Assuring Safety under Uncertainty in Learning-Based Control Systems

Thesis by

Richard Cheng

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2021
Defended December 15, 2020

Richard Cheng
ORCID: 0000-0001-8301-9169

ACKNOWLEDGEMENTS

I am deeply grateful for the guidance of my advisor Joel Burdick, who has been an extremely patient and caring advisor. His trust in me has given me the confidence to move forward in the most challenging periods of graduate school. I also want to thank my co-advisor Richard Murray, who has fundamentally shaped my way of thinking and doing research. I am grateful to him for always pushing me intellectually at critical junctures of my academic journey. I also want to thank Aaron Ames and Yisong Yue for being great mentors during my PhD and serving on my thesis committee.

I want to thank everyone from the Burdick group, AMBER lab, and the NCS group, who have been a significant source of inspiration as well as a great sounding board. My collaborators have also been critical to my growth over the past 5 years; in particular, I'd like to thank Yanan Sui, who provided me with much guidance and support early in my PhD, as well as Abhinav Verma, Ellen Novoseller, Daniel Pastor Moreno, and many others who helped me develop and improve my ideas.

The PhD journey has had many highs and lows, and I am grateful to all of my friends who celebrated the highs with me and kept me going during the lows. I must thank my officemates in GT 235 who have often made the office exciting and fun (and only slightly less productive). I also want to express my gratitude to all my Caltech friends across MCE, GALCIT, and CMS, as well as my Princeton "frands", who helped me put a pause on my PhD research whenever I needed to and brought me to love Pasadena.

I wouldn't be where I am today without my family. Mom, Dad, Kevin - thank you for the love you have shown me in the last 5 years, which helped me push through the toughest times. Last but not least, I'd like to thank my partner, Hanna, who has been a constant source of strength and support throughout the PhD.

ABSTRACT

Learning-based controllers have recently shown impressive results for different robotic tasks in well-defined environments, successfully solving a Rubiks cube and sorting objects in a bin. These advancements promise to enable a host of new capabilities for complex robotic systems. However, these learning-based controllers cannot yet be deployed in highly uncertain environments due to significant issues relating to learning reliability, robustness, and safety.

To overcome these issues, this thesis proposes new methods for integrating model information (e.g. model-based control priors) into the reinforcement learning framework, which is crucial to ensuring reliability and safety. I show, both empirically and theoretically, that this model information greatly reduces variance in learning and can effectively constrain the policy search space, thus enabling significant improvements in sample complexity for the underlying RL algorithms. Furthermore, by leveraging control barrier functions and Gaussian process uncertainty models, I show how system safety can be maintained under uncertainty without interfering with the learning process (e.g. distorting the policy gradients).

The last part of the thesis will discuss fundamental limitations that arise when utilizing machine learning to derive safety guarantees. In particular, I show that widely used uncertainty models can be highly inaccurate when predicting rare events, and examine the implications of this for safe learning. To overcome some of these limitations, a novel framework is developed based on assume-guarantee contracts in order to ensure safety in multi-agent human environments. The proposed approach utilizes contracts to impose loose responsibilities on agents in the environment, which are learned from data. Imposing these responsibilities on agents, rather than treating their uncertainty as a purely random process, allows us to achieve both safety and efficiency in interactions.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Richard Cheng, Krishna Shankar, and Joel W Burdick. "Learning an Optimal Sampling Distribution for Efficient Motion Planning". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020. URL: https://ras.papercept.net/proceedings/IROS20/1807.pdf.
R.C developed the algorithm, implemented and analyzed the method, conducted the experiments, and wrote the manuscript.

[2] Richard Cheng et al. "Safe multi-agent interaction through robust control barrier functions with learned uncertainties". In: *IEEE 59th Conference on Decision and Control (CDC)*. IEEE. 2020. URL: https://arxiv.org/abs/2004.05273.
R.C developed the algorithm, implemented and analyzed the method, conducted the simulations, and wrote the manuscript.

[3] Maegan Tucker et al. "Human Preference-Based Learning for High-dimensional Optimization of Exoskeleton Walking Gaits". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020. URL: https://arxiv.org/abs/2003.06495.
R.C participated in development of the learning algorithm and helped conduct human experiments.

[4] Richard Cheng et al. "Control Regularization for Reduced Variance Reinforcement Learning". In: *International Conference on Machine Learning*. 2019, pp. 1141–1150. URL: http://proceedings.mlr.press/v97/cheng19a.html.
R.C developed the algorithm, implemented and analyzed the method, conducted many of the simulations, and wrote the manuscript.

[5] Richard Cheng et al. "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 3387–3395. URL: https://www.aaai.org/ojs/index.php/AAAI/article/view/4213/4091.
R.C developed the algorithm, implemented and analyzed the method, conducted the simulations, and wrote the manuscript.

[6] Richard Cheng et al. "Motor control after human SCI through activation of muscle synergies under spinal cord stimulation". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 27.6 (2019), pp. 1331–1340. URL: https://ieeexplore.ieee.org/abstract/document/8704925.
R.C developed the muscle synergy extraction method, conducted the data analysis, and wrote the manuscript.

[7] Richard Cheng and Joel W Burdick. “Extraction of muscle synergies in spinal cord injured patients”. In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2018, pp. 2623–2626. URL: https://ieeexplore.ieee.org/abstract/document/8512763.
R.C developed the muscle synergy extraction method, conducted the data analysis, and wrote the manuscript.

[8] Richard Cheng et al. “On Muscle Activation for Improving Robotic Rehabilitation after Spinal Cord Injury”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 798–805. URL: https://ieeexplore.ieee.org/abstract/document/8593973.
R.C conducted the data analysis and wrote the manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Number *Page*

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Motivation

Over the past couple decades, significant progress has been made in the realm of robot planning and control, with many successes in areas such as warehouse automation [54], autonomous driving [175], and robotic manipulation [75, 124]. Many recent advancements have been enabled by the advent of machine learning and in particular, deep neural networks. Their ability to accurately capture extremely complex mappings directly from data and their impressive generalization to unseen scenarios have led to their widespread study in robotic systems. Indeed, much high-profile progress has leveraged these tools to learn complex behaviors from data and/or simulations [80, 151]. For example, reinforcement learning has recently been used to solve a Rubiks cube using a multi-fingered hand [9], and train robotic arms to autonomously pick up and sort a wide variety of objects [94]. While these successes are still limited to highly structured and controlled environments, the hope is that one day, robotic systems will be able to *safely* and *reliably* learn and execute complex behaviors in uncertain, unstructured environments (e.g. a crowded mall/street).

However, while there is great promise in leveraging machine learning approaches to enable new robotic capabilities directly from data or interactions, significant roadblocks still prevent widespread adoption. One of these roadblocks, which is the focus of this thesis, is certifying safety and improving reliability of learning-based control systems. While many recent works have shown impressive results using reinforcement learning to accomplish difficult tasks, widespread adoption will require much higher system reliability and formal guarantees of safety. For example, in the Rubiks cube example mentioned above, the robotic hand succeeded in solving the Rubiks cube 20% of the time [9]; in the sorting example, the system made a successful grasp 96% of the time [94]. In the self-driving car space, there have been notable high-profile safety failures, leading to significant injury and/or death, which have significantly damaged public trust. Therefore, while these previous advancements are extremely impressive, they are not sufficient to enable widespread deployment. In most applications, a robot that fails 1% of the time

will be frustrating to use; a robot that causes harm 0.01% of the time will not be acceptable. Therefore, the goal of this thesis is to explore how to leverage the high performance of reinforcement learning (and machine learning in general), while overcoming issues related to poor reliability and lack of safety guarantees arising from their black-box nature.

1.2 Problem Statement

The work in this thesis aims to improve reliability and guarantee safety in learning-based robotic control systems operating in uncertain environments, such that robots can be confidently deployed in more unstructured environments. This thesis examines two sub-areas of this problem, which are detailed below.

1.2.1 Ensuring Safety and Reliability in Reinforcement Learning

Reinforcement learning (RL) is a general technique aimed at finding an agent's optimal policy (i.e. controller) that maximizes long-term accumulated reward. In RL, the agent repeatedly observes its state/environment, takes an action according to its current policy, and receiving a reward. Over many iterations, the agent is able to modify its policy based on its experiences in order to maximize its long-term reward [161]. This method has seen recent success when applied to different robotic control tasks, learning to stabilize/operate complex robots in both simulation and the real world [9, 109, 148]. The appeal of reinforcement learning is that it can enable computation of controllers directly from high-dimensional observations and interactions with the environment, bypassing the need for accurate models or hand-crafted controller design.

However, these methods are also known to be unreliable (failing in many instances of learning) [136] and unsafe [13]. This stems from the mostly black-box treatment of most RL policies, making them uninterpretable, difficult to provide formal guarantees for, and highly susceptible to misspecified rewards. These difficulties raise mistrust of RL systems, and slow their adoption, regardless of the impressive tasks they may be able to accomplish. This has sparked significant interest in interpretable and safe RL (see Section 1.3 for an overview of such works).

In this thesis, I argue that some model information (i.e. prior knowledge of the robot and/or environment) is necessary to providing formal guarantees for and improve reliability of learning-based control systems. The thesis illustrates *how* to incorporate limited model information effectively into the model-free RL framework in order to improve learning reliability (Chapter 3), provide formal safety guarantees (Chapter

4), and guide the exploration process, thereby improving sample complexity. As opposed to most model-based RL methods (see Section 1.3.1), this work advocates for using model information to only constrain the action search space (rather than utilizing model-based learning updates). This allows us to leverage many of the benefits of model-based RL without being as susceptible to model inaccuracies.

1.2.2 Assuring Safety under Learned Models of Human Behavior

The latter half of this thesis focuses on assuring robot safety in multiagent *human* environments. As there is significant interest in adopting robots into many human environments, there has been much recent work looking at how to *(a)* learn accurate models of human behavior in complex, multiagent environments (see Section 1.3.3), and *(b)* provide probabilistic guarantees of safety using these learned behavior models (see Section 1.3.4). Much progress has been made in recent years on both of these fronts, in large part due to improved modeling using deep learning paired with large datasets, which can provide prediction of distributions over future human behavior. Therefore, this work studies how these models can be utilized to guarantee safety in safety-critical applications, and how/why they currently fail to do so.

While this topic has significant overlap with the previous topic (ensuring safety in RL), there is a fundamental difference between human uncertainty and any other types of uncertainty (e.g. dynamic uncertainty), which will be explored in depth. However, most works dealing with safety guarantees in human interaction do not make this distinction, which I argue leads to inaccurate models and suboptimal plans.

The first half of Chapter 5 examines different uncertainty models used to predict human behavior and guarantee safety in multiagent settings; it discusses how and why these models fall short in safety-critical applications, due to the unique nature of human uncertainty. The second half of Chapter 5 explores how to leverage and encode prior knowledge about human behavior and intention in order provide safety guarantees in multiagent, human settings based on more accurate, interpretable modeling assumptions.

1.3 Related Work

1.3.1 Reinforcement Learning for Planning and Control

In recent years, several works have studied reinforcement learning applied to continuous control tasks, leading to impressive results in both simulation and the real world [74, 109, 147]. As RL is a very general technique for solving the above

problem, there are many different classes of algorithms that fall under its umbrella. Most algorithms used for control tasks can be classified into two categories: (1) model-free RL, and (2) model-based RL.

Model-Free Reinforcement Learning: Model-free RL refers to approaches where the policy (most often parameterized by a neural network) is learned/optimized based on observed experiences. Typically, these approaches learn a value function, directly learn a policy, or both. Many works have shown that learning a value function (i.e. a complex function that represents the long-term value of taking any action) can lead to very good planners (e.g. Q-learning, SARSA), particularly in the discrete action domain [43, 137]. More recently, many of these methods have been combined with deep neural networks (to parameterize the value function), and have shown impressive results with the resulting planners [47, 120]. However, the problem of synthesizing a controller from an accurate Q-function is non-trivial.

Therefore, policy gradient methods have become extremely popular for continuous control tasks, as they allow direct optimization of the policy/controller (rather than having to go through a value function). In one of the earliest formulations of the policy gradient, Williams [178] showed that an unbiased estimate of the policy gradient could be directly and easily computed from sampled data/trajectories. Therefore, in a given learning episode, data would be gathered, the policy gradient could be computed, the policy would be updated; then new data would be gathered with the updated policy. Silver [150] introduced deterministic policy gradients for deterministic policies.

While elegant, these policy gradients exhibited extreme variance, limiting their usefulness [186]. However, it was found that this variance could be reduced without introducing bias by subtracting a baseline from the reward function in the policy gradient [79, 176], and the action-value function could provide an optimal baseline [24, 159]. Therefore, an improved policy gradient was formulated to include the action-value function [159]. However, this helped the variance problem, but introduced the new issue of getting a good value function estimate.

This gave rise to many well-performing actor-critic methods, where an actor represents the policy/controller for the system and the critic represents the estimate of a value function or some variant. These methods utilize the critic as a baseline for variance reduction for updates of the actor [146]. Alternating updates are made to both functions; while this may cause learning instabilities, such methods have led to many successful algorithms [84, 147, 148, 179].

As reinforcement learning methods struggle with sample complexity, many methods have attempted to boost data efficiency by saving data from previous learning iterations (e.g. in a "replay-buffer") [14, 109, 145]. These methods are referred to as off-policy methods, as they rely on data gathered based on a separate policy to update the current policy. In contrast, on-policy methods only rely on data gathered based on the current policy to update the current policy [147].

However, despite the successes of model-free reinforcement learning, many have noted that variance still remains problematically high in reinforcement learning, making learning unreliable [88, 90, 136]. Furthermore, while these techniques bypass the need for complex controller synthesis based on an accurate system model, they typically exhibit poor sample complexity [136].

Model-Based Reinforcement Learning: In contrast, model-based RL algorithms learn a model of the environment, which they leverage to learn the optimal policy (or value function) [122]. There are several different types of model-based RL algorithms, which depend on *(a)* how the environment model is learned, and *(b)* how that model is leveraged for planning.

The first part of a model-based method learns an environment model, which can be accomplished with a variety of methods, from linear regression to neural networks [171]. There are a wide variety of different dynamical system models, which allow us to predict future system states and potentially the uncertainty in these states (e.g. neural network with dropout [70], Gaussian process [44]) — see [122] for an extensive review.

Once this model has been learned, there are several approaches to utilizing it for planning. Most methods fall into three main categories:

- Use the model for approximate dynamic programming to obtain the system's value function [51, 107];
- Sample additional data from model, which can then be used for standard model-free updates [160];
- Use the model for gradient-based planning (e.g. compute policy gradient by differentiating through the dynamics model) [44, 184]. Other work uses Bayesian optimization for policy updates based on the learned model [21].

A more extensive review on these methods and model-based reinforcement learning can be found in [121]. While model-based RL methods enable better sample complexity, model accuracy has a significant impact on the learning task, making performance susceptible to slight model errors.

Hybrid Methods: In this thesis, the main focus is on ways to intelligently incorporate model information into the model-free RL framework, in order to realize the benefits of model-free RL while overcoming its major limitations (i.e. safety, reliability, sample complexity). Along this line, work by [92, 152] adds a control prior during learning, and empirically demonstrates improved performance. Researchers in [59, 125] used model-based priors to produce a good *initialization* for their RL algorithm, but did not use regularization during learning. Another approach [106] uses a model-based controller to guide the robot to areas where the model is poor, where model-free learning takes over. Other works incorporate model information only to filter the learned policy to guarantee safety [65], but do not consider the impact of this filtering on learning. Discussion of these works is left to the following subsection on safe RL.

1.3.2 Safe Reinforcement Learning for Control

Safe RL is a subfield of reinforcement that focuses on learning a policy that maximizes the expected return, while also ensuring (or encouraging) the satisfaction of some safety constraints [71]. Model-free approaches to safe reinforcement learning include *policy optimization with constraints* [6, 72, 123, 170], or *teacher advice* [4, 5, 163]. These methods have demonstrated that they can quickly learn to satisfy specified constraints after a short period of learning. However, these model-free approaches do not guarantee safety during learning – safety is only approximately guaranteed after a sufficient learning period (e.g. guaranteed in expectation [6]). The fundamental issue with model-free safe reinforcement learning is that without a model, safety must be learned through environmental interactions, which means it may be violated during those interactions.

Model-based approaches have utilized Lyapunov-based methods [25, 38, 133] or model predictive control [99] to guarantee safety under system dynamics during learning. Bayesian inference was utilized in [56, 97, 131, 173] to learn system dynamics in an online manner while ensuring probabilistic safety using Control Barrier Functions. However, these works do not consider the issue of RL exploration and performance optimization. They capture uncertainty in the system dynamics

(typically using a Gaussian model), and design a controller that is robust to this uncertainty. Similarly, recent methods rely on Hamilton-Jacobi reachability in order to incorporate model information into model-free RL algorithms to ensure safety during exploration [65, 76]. These approaches directly filter the control action of the RL controller to guarantee safety, but do not consider interaction between the learner and the model-based safety-filter, which can distort the learning process (as will be discussed in Chapter 4). Short-term predictions and rule-based priors can also constrain the action set [119]. Some works have looked at guaranteeing safety by switching between backup controllers [115, 133]. It is also possible to add a collision penalty in the objective based on the model's predicted probability (and uncertainty) of collision [93] .

The work in this thesis relies heavily on control barrier functions (CBFs) for safety assurances [11, 12, 128] (see Section 2.2 for technical background). CBFs use Lyapunov-like arguments to ensure safety through set invariance, but they are more suited to safety requirements as they deal with set stability rather than point-wise stability. They have been used to guarantee collision avoidance in multiagent settings by projecting desired actions, to the closest (in least-squares sense) safe actions according to a CBF condition [39, 156, 182]. Recent works have looked at learning CBFs for general systems, either implicity or explicitly [83, 157]. A multi-agent CBF was defined explicitly for multi-agent systems in the case of continuous-time linear dynamics [27, 172]. While most work in this area has focused on continuous-time dynamics, some recent work has also examined discrete-time control barrier functions [7, 8], which will be leveraged in this thesis.

While CBFs enable formal guarantees of safety for dynamical systems in a computationally efficient manner, most works rely on a known model of the system dynamics. However, given the need to deal with environmental uncertainty, many recent works have looked at developing robust control barrier functions or incorporating uncertainty into the CBF formulation for probabilistic safety guarantees [37, 114, 153, 154].

1.3.3 Uncertainty Modeling in Safe Control

Almost all approaches for guaranteeing safe control in the presence of uncertainty rely on an assumed or learned model that approximates that uncertainty as a random process (e.g. deviations from a nominal dynamics model are drawn i.i.d. from some probability distribution). These uncertainty models help capture noise and

the effects of unobserved variables, and they enable probabilistic safety guarantees by placing bounds on the uncertainty that our agent may face and must be robust to. This section provides an overview of these uncertainty models, which can almost all be placed into one or more of the following categories:

- **Gaussian Process (GP):** These approaches model other agents' trajectories as Gaussian processes, which treat trajectory uncertainty as a multivariate Gaussian [15, 37, 53, 86, 140]. There are several extensions, such as the IGP model [165] (which accounts for interaction between multiple agents), or others [60, 110]. However, they all treat uncertainty as a multivariate Gaussian.

- **Gaussian Noise with Dynamics Model:** These approaches use a dynamics model with additive Gaussian noise; noise can also be added in state observations. This induces a Gaussian distribution over other agents' future trajectory (or a situation amenable to moment-matching) [68, 78].

- **Quantile Regression:** This approach computes quantile bounds over the trajectories of other agents at a given confidence level, δ. This approach is beneficial in that it does not assume an uncertainty distribution over trajectories [57, 162].

- **Scenario Optimization:** This approach computes a predicted set over other agents' actions based on samples of previously observed scenarios [30]. This is a distribution-free approach (i.e. does not assume a parametric uncertainty distribution) [31, 32, 36, 144].

- **Noisy Rational (i.e. Boltzmann Rational) model:** This model treats the human as a rational actor who takes "noisily optimal" actions according to a distribution in the exponential family, shown in Equation (5.5). The uncertainty in the action is captured by this distribution, which relies on an accurate model of the human's value function [63, 69, 102, 108, 141].

- **Generative Models (e.g. CVAE, GAN):** These models learn an implicit distribution over trajectories. They do not provide an explicit distribution, but can generate random trajectories that attempt to approximate the true distribution [81, 138, 143].

- **Hidden Markov Model (HMM) / Markov Chain:** These models differ in that they capture uncertainty over *discrete* sets of states/intentions (e.g. goal positions) – as opposed to capturing uncertainty over trajectories. Thus, the objective is to infer the other agents' unobserved state/intention (from a discrete set) with very high certainty, $1 - \delta$ [19, 95, 104, 111, 116, 139, 164].
- **Uncertainty Quantifying (UQ) Neural Networks:** These approaches do not constitute a separate class of uncertainty models, but simply refer to methods that train a neural network to capture the distribution over other agents' trajectories [77, 93, 117]. They are listed separately due to their popularity. Most often these networks output a Gaussian distribution or mixture of Gaussians (e.g. Bayesian neural networks [26], deep ensembles [103], Monte-Carlo dropout [70]). These models can also quantify uncertainty over discrete states (i.e. infer the hidden state in HMMs) [46, 89].

Among these, the Gaussian uncertainty model is the most popular, and most uncertainty-quantifying neural networks that aim to learn some dynamic uncertainty will output a Gaussian distribution or GMM. Once an uncertainty class is chosen, and the uncertainty is learned from data, it can be incorporated into one of several safety mechanisms that exist to guarantee safety (e.g. integrating uncertainty into chance constraints).

1.3.4 Assuring Safety in Human Multi-Agent Scenarios

Multi-agent collision avoidance has been a long-studied problem with different approaches proposed for enabling safe control in varying situations. Velocity obstacles is a popular approach that involves limiting control actions to a set of "safe" actions, though its assumption of constant velocity with linear dynamics is limiting [62, 167]. Related works in this direction have loosened these assumptions, but require significant sampling of the action space and do not incorporate dynamic uncertainty [22, 177]. More recently, Buffered Voronoi Cells (BVC) have been proposed as a tool to provide safety guarantees with only positional information, though safety guarantees are provided only under linear dynamics and without uncertainty [174, 187]. Also, [20, 34, 67] provide safety guarantees, under worst case disturbances, by solving the Hamilton-Jacobi-Isaacs equation to obtain a minimally invasive control law. However, the heavy computational expense prohibits applicability to large-scale multi-agent systems.

Reinforcement learning methods have emerged recently for multiagent planning/control, which directly learn actions in multi-agent settings in response to the observed environment [35, 55]. However, these methods provide no formal guarantees of safety, and as such are prone to collision in novel environments. Furthermore, they have not been shown to scale to settings with many agents. Lotjens [112] provides safety RL in the presence of humans, but only considers epistemic model uncertainty with collision probability penalized in the objective function. Zhang [185] combined a safe backup policy with the learned policies. Recent work has also looked at generative models in order to predict other agents' future trajectories and encourage safety with respect to them, and these generative models have shown promising results though they lack formal safety guarantees [130].

Another approach treats the planning problem as a Partially Observable Markov Decision Process (POMDP), where the goal or intention of the other agents is unobserved. By considering a finite set of goals/intentions, the goal/intention can be probabilistically inferred [19, 29, 42, 118]. Some works have integrated this with the safe RL framework [28]. Such a formulation is nice, because it naturally incorporates human intention (and its uncertainty) into the MDP framework.

Methods based on inverse reinforcement learning (IRL), which aims to learn other agents' reward function in order to implicitly understand their goals, have also shown promise for safe human-robot interaction [188]. These IRL methods are loosely inspired by psychology studies suggesting that humans behave (noisily) optimally with respect to some metric [41, 45]. After learning this reward function, a game can be solved between the primary agent to be controlled and the other agent(s) in order to compute the optimal trajectory [18, 50, 85, 141, 142]. Many of these works model interaction as a Markov Game, while others [66, 108, 183] utilize a Stackelberg dynamic game. The game-theoretic formulation is a nice mathematical formulation in that it allows us to consider optimality of the entire community, and allows us to leverage the influence one agent's actions have on others. [69] has shown that safety guarantees can be made by considering that the uncertainty obeys the "noisy rational" model of the human. Other recent work has considered risk awareness under a set of reward functions [102].

1.4 Contributions and Structure of this Thesis

The main contributions of this thesis can be summarized as follows:

- I develop a method for integrating a model-based controller into the reinforce-

ment learning framework, and show (both theoretically and empirically) that this significantly reduces variance in learning with minimal bias introduced. This approach is proven to be equivalent to constraining the policy search space, and it is also proven that robustness properties of the control prior can be inherited by the RL policy.

- I develop a framework that utilizes limited model information to guarantee safety by leveraging discrete-time control barrier functions and Gaussian process uncertainty models. It is shown that policy gradient distortion occurs under naive inclusion of these control barrier functions (or any safety filter), and this work proposes a method that accounts for the policy gradient distortion. I prove that this method both guides the exploration process of learning and allows for probabilistic safety guarantees.

- An in-depth analysis is conducted, evaluating the accuracy of major uncertainty models in predicting the distribution of human behavior. In particular, by examining human driving behavior, it is shown that all these uncertainty models perform very poorly (exhibiting very low accuracy) when predicting the probabilities of rare events. It is further shown that the amounts of data required to obtain accurate predictions for the probabilities of these rare events is currently infeasible.

- To overcome the issues associated with inaccurate uncertainty models, a new method based on assume-guarantee contracts is proposed for guaranteeing safety in human environments. A framework is developed for learning the main components of these contracts, and it is proven that under certain assumptions on the behavior of agents, safety is always guaranteed when these contracts are satisfied. This also enables us to assign responsibility for safety violations by examining contract violations.

Chapter 2 provides technical background on topics related to the major themes of this thesis. Chapters 3 and 4 will focus on the issues of reliability and safety, respectively, in deep reinforcement learning. Chapter 3 dives into the extreme variance that results from inevitable random exploration in reinforcement learning, and how it can be alleviated by using limited model information, while also improving robustness and sample complexity. Chapter 4 focuses on providing probabilistic guarantees of safety in reinforcement learning through the use of and how this can also guide the exploration/learning process. The first half of Chapter 5 conducts an analysis

into the uncertainty models used to guarantee safety throughout the literature, and shows that for safety-critical robotic applications (e.g. self-driving), these models are highly inaccurate and cannot provide reliable safety guarantees. The second half of Chapter 5 introduces a novel planning approach based on assume-guarantee contracts that attempts to deal with the aforementioned inaccuracy in uncertainty models.

Chapter 2

BACKGROUND AND PRELIMINARIES

This chapter introduces the main concepts and mathematical background that underpins the rest of this thesis.

Let us model the system of interest by an infinite-horizon discounted Markov decision process (MDP) with dynamics defined by the tuple $(S, A, f^{(a)}, r, \gamma)$, where $S \subseteq \mathbb{R}^n$ is the state space, $A \subseteq \mathbb{R}^m$ is the action space, $f^{(a)} : S \times A \to S$ describes the nonlinear system dynamics (which may be unknown to the agent), $r(s, a) : S \times A \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. The evolution of the system is given by the following dynamical system and its continuous-time analogue,

$$\begin{aligned} s_{t+1} &= f^{(a)}(s_t, a_t), \\ \dot{s} &= f_c^{(a)}(s, a), \end{aligned} \tag{2.1}$$

where $\dot{s}$ denotes the continuous time-derivative of the state s, and $f_c^{(a)}(s, a)$ denotes the continuous-time analogue of the discrete time dynamics $f^{(a)}(s_t, a_t)$.

2.1 Reinforcement Learning

Define a stochastic policy (i.e. controller) $\pi_\theta(a|s) : S \times A \to [0, 1]$ parameterized by θ. The overarching aim of RL is to find the policy (i.e. parameters, θ) that maximizes the expected accumulated reward, $J(\theta)$, for the MDP defined above, where

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]. \tag{2.2}$$

Here, $\tau \sim \pi_\theta$ is a trajectory $\tau = \{s_t, a_t, ..., s_{t+n}, a_{t+n}\}$ whose actions and states are sampled from the policy distribution $\pi_\theta(a|s)$ and the dynamics (2.1), respectively.

Under the reinforcement learning umbrella, there are a diverse set of methods that have been proposed for learning controllers from interactions with the environment. An overview of such methods was provided in Section 1.3.1, but for now let us focus on policy gradient methods (PGM), as they are one of the most popular tools for learning control tasks. PGMs will be leveraged extensively in Chapters 3 and 4.

The policy gradient, $\nabla_\theta J(\pi_\theta)$, provides a very simple way to iteratively update/optimize the policy π_θ from one learning iteration to the next:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_\theta). \tag{2.3}$$

The challenge is that $\nabla_\theta J(\pi_\theta)$ is notoriously difficult to accurately estimate/compute. In 1992, Williams [178] showed that the following expression represents an unbiased estimator for the policy gradient,

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\nabla_\theta \log \pi_\theta(\tau) R(\tau)\right] \tag{2.4}$$

where $R(\tau) = \sum_{t=0}^{\infty} r(s_t, a_t)$. Intuitively, this gradient can be viewed as simply updating the policy parameters, θ, to increase the probability of taking actions that led to higher reward and vice versa. However, though (2.4) can be used to compute an unbiased estimate of the policy gradient, it was found to have extremely high variance, leading to poor performance in practice.

To address this issue, it was shown that the reward $R(\tau)$ can be replaced with the action-value function, Q^{π_θ} [159] (or the advantage function [148]), which reduces the variance in the policy gradient estimates. Therefore, the policy gradient can be estimated as,

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\nabla_\theta \log \pi_\theta(\tau) Q^{\pi_\theta}\right], \tag{2.5}$$

where the action-value function, value function, and advantage function are defined as,

$$\begin{aligned} Q^{\pi}(s_t, a_t) &= \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots}\Big[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l})\Big], \quad a_j \sim \pi(s_j) \\ V^{\pi}(s_t) &= \mathbb{E}_{a_t, s_{t+1}, a_{t+1}, \ldots}\Big[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l})\Big], \quad a_j \sim \pi(s_j) \\ A_\pi(s_t, a_t) &= Q_\pi(s_t, a_t) - V_\pi(s_t). \end{aligned} \tag{2.6}$$

With a perfectly accurate action-value function Q^{π_θ}, (2.5) is an unbiased estimator; with a reasonably accurate Q^{π_θ}, (2.5) is a low-bias estimator. Therefore, while the action-value function helps reduce the variance in the policy gradient, it introduces its own problem, which is: how does one estimate Q^{π_θ} (especially when it depends explicitly on π_θ)? This gives rise to the widely used/studied actor-critic methods [84, 147].

Remark 1. Different algorithms may replace Q^{π_θ} in (2.5) with a different quantity (e.g. the advantage function [146]) for variance reduction. However, the policy gradient concept remains unchanged.

Let us refer to the policy π_θ as the "actor" and the action-value function Q^{π_θ} as the "critic." Actor-critic methods alternately make incremental updates to the actor π_θ, using the policy gradient (2.5) (or some variant of it), and then the critic Q^{π_θ}. The critic updates are typically made using temporal-difference error or Monte-Carlo estimates, though examples of other variants can be found in [161]. Some of the most impressive results in reinforcement learning for control have been achieved by leveraging actor-critic methods. However, due to the tight dependency between the actor and critic, the update step sizes must be chosen carefully to avoid learning instability. Furthermore, these methods still suffer from poor reliability and can often exhibit unsafe behavior, which will be the main focus of Chapters 3 and 4.

2.2 Control Barrier Functions

Control barrier functions are a useful method used to ensure safety in nonlinear control systems by guaranteeing set invariance [11]. Let us consider an arbitrary safe set, C, defined by the super-level set of a smooth function $h : \mathbb{R}^n \to \mathbb{R}$,

$$C : \{s \in \mathbb{R}^n : h(s) \geq 0\}. \tag{2.7}$$

System safety is maintained if and only if the system state, s, remains within the safe set C for all time (i.e. the set C is *forward invariant* under the flow of the controlled system). Examples include keeping a manipulator within a given workspace, or ensuring that a robot avoids obstacles. Control barrier functions utilize a Lyapunov-like argument to provide a sufficient condition for ensuring forward invariance of the safe set C under controlled dynamics, and are therefore a natural tool to enforce safety and synthesize safe controllers [11].

Definition 1. Given a set $C \in \mathbb{R}^n$ defined by (2.7), the smooth function $h : \mathbb{R}^n \to \mathbb{R}$ is a *discrete-time control barrier function* (CBF) for dynamical system (2.1) if there exists $\eta \in [0, 1]$ such that for all $s_t \in C$,

$$\sup_{a_t \in A} \left[h\Big(f^{(a)}(s_t, a_t)\Big) + (\eta - 1)h(s_t) \right] \geq 0, \tag{2.8}$$

where η represents how strongly the control barrier function "pushes" the state inwards within the safe set.

For convenience, let us define the *control barrier condition* (CBC) corresponding to the CBF, h, as follows,

$$h\Big(f^{(a)}(s_t, a_t)\Big) + (\eta - 1)h(s_t) \geq 0, \tag{2.9}$$

The existence of a CBF implies that there exists a *deterministic* controller $u^{CBF} : S \to A$ such that the set C is forward invariant with respect to the system dynamics (2.1) [7, 12]. Given a CBF and known system dynamics, one can synthesize such a controller by formulating and solving an optimization problem that enforces the CBC (2.9) as follows,

$$\begin{aligned} a_t = \underset{a \in A}{\text{argmin}} \quad & \|a\|_2 \\ \text{s.t.} \quad & h\big(f^{(a)}(s_t, a)\big) + (\eta - 1)h(s_t) \geq 0 \\ & a^i_{low} \leq a^i \leq a^i_{high} \ \text{ for } i = 1, ..., m \end{aligned} \tag{2.10}$$

where the constraints on a^i_t encode actuator limits. The solution to this optimization problem implicitly defines a safe controller u^{CBF}, since it satisfies the CBC (2.9) for all $s \in C$, thereby rendering C forward invariant. One of the goals in Chapter 4 will be to efficiently incorporate CBF-based controllers into the RL framework in order to both improve learning efficiency and guarantee safety.

Multi-agent CBF: In general, a valid control barrier function can be difficult to compute and is highly problem-specific. However, for multi-agent, continuous-time linear systems, Borrman et al. [27] proposed the following CBF,

$$h(s) = \frac{\Delta p^T \Delta v}{\|\Delta p\|} + \sqrt{2u_{max}(\|\Delta p\| - D_s)}\,, \tag{2.11}$$

where u_{max} represents each robot's maximum acceleration, D_s is the collision margin, Δp is the positional difference between agents, and Δv is the velocity difference between agents. Chapter 4.4 will extend this multiagent CBF to discrete-time systems with potentially non-linear control-affine dynamics.

Remark 2. In general, the nonlinear optimization problem (2.10) is difficult to solve in a computationally efficient manner, especially when the dynamics $f^{(a)}$ are complex and uncertain. Chapter 4 will discuss some of the simplifications/assumptions made in the CBF literature that can simplify the optimization (2.10) to a quadratic program, and propose a method for achieving this with multi-agent CBFs (2.11).

2.3 Gaussian Process Models

Any controller that provides safety guarantees under uncertainty must capture/approximate that uncertainty in some way. While there are several different models of uncertainty, each with their own benefits and drawbacks, this thesis will draw heavily on the Gaussian process model (see Section 1.3.3 for a summary of other uncertainty models).

A Gaussian process (GP) is a nonparametric regression method for estimating functions *and their uncertain distribution* from data or observations [135]. For example, suppose our robot dynamics are 1-dimensional and uncertain, such that the dynamics are described by $s_{t+1} = d(s_t)$, where d is unknown. Then a GP can provide an evolving model of the uncertain function $d(s)$, by a mean estimate function, $\mu_d(s)$, and an uncertainty function, $\sigma_d^2(s)$. The GP model is able to do this by treating every set of observations as being drawn from a multivariate Gaussian random variable. Therefore, the model is fully specified by a kernel function $k(s, s')$, which defines the similarity between any two states $s, s' \in S$. A common kernel function, and the one that is primarily used in this thesis, is the squared exponential kernel,

$$\kappa(s_i, s_j) = \sigma^2 \exp\left(\frac{-\|s_i - s_j\|^2}{2l^2}\right), \tag{2.12}$$

where σ and l are kernel hyperparameters. However, there are many possible choices of kernel functions, and more details about them can be found in [135].

Given this kernel function κ, one can infer a distribution over the function $d(s)$ based solely on previous measurements of that function, $\hat{d}(s)$. Suppose that N measurements $d_{[N]} := [\hat{d}(s_1), \hat{d}(s_2), ..., \hat{d}(s_N)]$ are taken at sampling points $s_{[N]} := [s_1, \ldots, s_N]$, subject to independent Gaussian noise $\nu_{noise} \sim \mathcal{N}(0, \sigma^2_{noise})$. Since any finite number of data points form a multivariate normal distribution in the GP, then the following holds,

$$\begin{bmatrix} d_{[N]} \\ d(s_*) \end{bmatrix} = \mathcal{N}\big(0, \begin{bmatrix} K(s_{[N]}, s_{[N]}) + \sigma^2_{noise} I & K(s_{[N]}, s_*) \\ K(s_*, s_{[N]}) & \kappa(s_*, s_*) + \sigma^2_{noise} \end{bmatrix}\big) \tag{2.13}$$

where $K(s_{[N]}, s_{[N]}) \in \mathbb{R}^{N \times N}$ with $[K(s_{[N]}, s_{[N]})]_{i,j} = \kappa(s_i, s_j)$, and $K(s_*, s_{[N]}) \in \mathbb{R}^{1 \times N}$ with $[K(s_*, s_{[N]})]_i = \kappa(s_*, s_i)$, and σ^2_{noise} is the variance of additive measurement noise. Based on (2.13), one can then obtain the posterior distribution of $d(s_*)$ at any query state $s_* \in S$ by conditioning on the past measurements. The mean $\mu_d(s_*)$ and variance $\sigma_d^2(s_*)$ of the function d at the query state, s_*, are calculated as,

$$\begin{aligned} \mu_d(s_*) &= K^T(s_{[N]}, s_*)(K(s_{[N]}, s_{[N]}) + \sigma^2_{noise} I)^{-1} d_{[n]}, \\ \sigma_d^2(s_*) &= k(s_*, s_*) - K^T(s_{[N]}, s_*)(K(s_{[N]}, s_{[N]}) + \sigma^2_{noise} I)^{-1} K(s_{[N]}, s_*), \end{aligned} \tag{2.14}$$

With this model of the posterior distribution over $d(s)$, high probability confidence intervals on the function can be easily computed,

$$\mu_d(s) - k_\delta \sigma_d(s) \leq d(s) \leq \mu_d(s) + k_\delta \sigma_d(s), \tag{2.15}$$

with probability $(1-\delta)$ where k_δ is a design parameter that determines δ. Therefore, by learning $\mu_d(s)$ and $\sigma_d(s)$ in tandem with the controller, high probability confidence intervals on the unknown dynamics can be found, and these intervals adapt/shrink as more information (i.e. measurements) on the system is obtained. With more collected data, $\mu_d(s)$ becomes a better estimate of $d(s)$, and the uncertainty, $\sigma_d^2(s)$, of the dynamics decreases.

Matrix-Variate Gaussian Processes: The GP model described above infers the posterior over 1-dimensional functions. This limitation is easily overcome by utilizing a separate GP model for each dimension of the output function, d [173]. However, this approach ignores correlations between the components of a multivariate uncertainty $d(s) \in \mathbb{R}^n$. To extend the model to the multivariate setting and account for potential correlations, the Matrix-Variate Gaussian Process (MVG) model can be utilized to infer a multivariate distribution over $d(s)$ from data. First, let us define the MVG distribution [82, 97, 113, 158] over a random variable S by the following probability density function,

$$p(\mathbf{S}; \mathbf{M}, \Sigma, \Omega) := \frac{\exp\left(-\frac{1}{2}\,\mathbf{tr}\left[\Omega^{-1}(\mathbf{S}-\mathbf{M})^\top \Sigma^{-1}(\mathbf{S}-\mathbf{M})\right]\right)}{(2\pi)^{nm/2}\det(\Sigma)^{n/2}\det(\Omega)^{N/2}}, \tag{2.16}$$

where $M \in \mathbb{R}^{N\times n}$ denotes the mean, and $\Sigma \in \mathbb{R}^{N\times N}$ encodes the covariance matrix of the rows, and $\Omega \in \mathbb{R}^{n\times n}$ encodes the covariance matrix of the columns. In this case, one can write $\mathbf{S} \sim \mathcal{MN}(\mathbf{M}, \Sigma, \Omega)$, or equivalently $vec(\mathbf{S}) \sim \mathcal{N}(vec(\mathbf{M}), \Sigma \otimes \Omega)$, where $vec(\mathbf{S}) \in \mathbb{R}^{nm}$ is the vectorization of S, obtained by stacking the columns of S, and $\otimes$ is the Kronecker product.

Without loss of generality, assume a zero mean function for the MVG with positive semi-definite parameter covariance matrix $\Omega \in \mathbb{R}^{n\times n}$, and kernel $\kappa : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ (e.g. the squared exponential kernel (2.12)). Therefore, in order to model $d(s)$ with the MVG, let us assume the following,

$$vec(d(s_1), \ldots, d(s_N)) \sim \mathcal{N}(\mathbf{0},\ \Sigma(s_{[N]}) \otimes \Omega), \tag{2.17}$$

where $\Sigma \in \mathbb{R}^{N\times N}$ with $\Sigma_{i,j} = \kappa(x_i, x_j)$. As in the univariate GP case, the training observations $d_{[N]}$ at sampling points $s_{[N]}$ and the predictive target $d(s_*)$ at query test point s_* are assumed to be jointly Gaussian such that,

$$\begin{bmatrix} d_{[N]} \\ d(s_*) \end{bmatrix} \sim \mathcal{MN}\left(\mathbf{0}, \begin{bmatrix} K(s_{[N]}, s_{[N]}) & K(s_*, s_{[N]})^T \\ K(s_*, s_{[N]}) & \kappa(s_*, s_*) \end{bmatrix}, \Omega\right), \tag{2.18}$$

where $K(s_{[N]}, s_{[N]}) \in \mathbb{R}^{N \times N}$ with $[K(s_{[N]}, s_{[N]})]_{i,j} = \kappa(s_i, s_j)$, and $K(s_*, s_{[N]}) \in \mathbb{R}^{1 \times N}$ with $[K(s_*, s_{[N]})]_i = \kappa(s_*, s_i)$. Thus, the posterior distribution over d can be computed as follows:

$$\begin{aligned} d(s_*) \sim & \mathcal{N}\left(\hat{M}\ ,\ \hat{\Sigma} \otimes \hat{\Omega}\right) \\ & \hat{M} = K(s_*, s_{[N]})^T K(s_{[N]}, s_{[N]})^{-1} d_{[N]} \\ & \hat{\Sigma} = \kappa(s_*, s_*) - K(s_{[N]}, s_*)^T K(s_{[N]}, s_{[N]})^{-1} K(s_{[N]}, s_*) \\ & \hat{\Omega} = \Omega \end{aligned} \tag{2.19}$$

Remark 3. The Gaussian process models every set of observations as being drawn from a multivariate (or matrix-variate) Gaussian random variable. This powerful assumption allows one to predict a distribution over $d(s)$ for some unobserved state s based on other observed states, $(s_1, ..., s_n)$. However, if this assumption is inaccurate, then the predicted distribution over $d(s)$ may also be inaccurate. The accuracy of these models is discussed in Chapter 5.1.

Remark 4. In applications involving large amounts of data, training the GP becomes problematic since computing the matrix inverse in Equations (2.14) and (2.19) scales poorly (typically N^3 in the number of data points). There are several methods to alleviate this issue, such as using sparse inducing inputs or local GPs [129, 155]. This work bypasses this issue by batch training the GP model with only the latest batch of data points of limited size.

2.4 Mathematical Notation

This section introduces miscellaneous mathematical notation that will be used. Throughout the thesis, u will denote deterministic controllers $u : S \to A$, while π will denote stochastic policies $\pi : S \times A \to [0, 1]$. Therefore, a controller, u, can be drawn from a policy π. Let $*$ be the convolution operator, which will be used to combine policies. If $u_1 \sim \pi_1$ and $u_2 \sim \pi_2$, then the combined controller $u_{tot} = u_1 + u_2$ can be seen as being drawn from $u_{tot} \sim \pi_1 * \pi_2$. Let $D_{KL}(p|q)$ and $D_{TV}(p|q)$ denote the Kullback-Leibler divergence, and total variation distance, respectively, between probability distributions p and q.

Chapter 3

CONTROL REGULARIZATION FOR REINFORCEMENT LEARNING

As discussed in Chapter 1, despite impressive demonstrations in recent years, model-free RL still faces significant issues with reliability, safety, and sample complexity. Complex tasks can take millions of iterations to learn. More importantly, the variation over learning runs can be very high, meaning some runs of an RL algorithm succeed while others fail, depending on randomness in initialization and sampling. Several studies have noted this high variability in learning as a significant hurdle for the application of RL, since learning becomes unreliable [17, 88, 136]. All policy gradient algorithms face this same issue, which is illustrated in Figure 3.1.

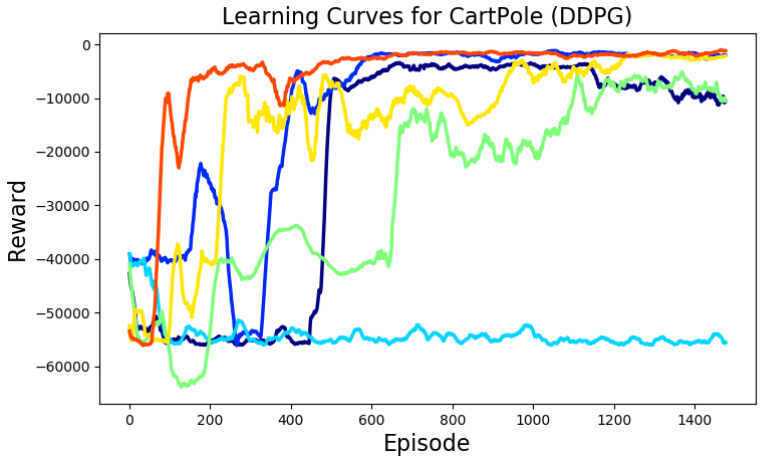


Figure 3.1: Learning curves for 6 separate learning trials when using the RL algorithm DDPG [150] to solve the CartPole task in the OpenAI gym environment. All hyperparameters are kept constant, with only the random seed changed between trials. However, significant variance is seen in learning performance, with one of the trials completely unable to improve on the task.

This chapter focuses on how to incorporate limited model information into the model-free RL framework in order to reduce learning variance, improve sample complexity, and avoid significant bias. In addition to provably reducing learning variance, I show that the proposed algorithm can also improve controller robustness in certain applications.

To illustrate the root of the variance issue inherent in reinforcement learning, recall the problem defined in Equation (2.2), which is to find the optimal policy that

maximizes an agent's long-term reward. In the learning problem, one must estimate the gradient of the expected return $J(\theta)$ with respect to the policy based on sampled trajectories. Recall from Section 2.1 that this gradient, $\nabla_\theta J$, can be estimated as,

$$\begin{aligned}\nabla_\theta J(\theta) = \; & \mathbb{E}_{\tau\sim\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(\tau) Q^{\pi_\theta}(\tau)\Big] \\ & \approx \sum_{i=1}^{N}\sum_{t=1}^{T}[\nabla_\theta \log \pi_\theta(s_{i,t}, a_{i,t}) Q^{\pi_\theta}(s_{i,t}, a_{i,t})].\end{aligned} \tag{3.1}$$

However, the estimated policy gradient has very high variance (even when using the action-value function Q^{π_θ}), because the expectation in (3.1) must be estimated using a finite set of sampled trajectories. This high variance in the policy gradient, $\text{var}[\nabla_\theta J(\theta_k)]$, translates to high variance in the updated policy, $\text{var}[\pi_{\theta_{k+1}}]$, as seen below,

$$\begin{aligned}&\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta_k), \\ &\pi_{\theta_{k+1}} = \pi_{\theta_k} + \alpha \frac{d\pi_{\theta_k}}{d\theta} \nabla_\theta J(\theta_k) + O(\Delta\theta^2), \\ &\text{var}[\pi_{\theta_{k+1}}] \approx \alpha^2 \frac{d\pi_{\theta_k}}{d\theta} \text{var}[\nabla_\theta J(\theta_k)] \frac{d\pi_{\theta_k}}{d\theta}^T \quad \text{for } \alpha \ll 1,\end{aligned} \tag{3.2}$$

where α is the user-defined learning rate. It is important to note that the variance of interest is with respect to the gradient estimate $\nabla_\theta J$ arising from randomness in the sampled trajectories.

3.1 Reducing Variance in RL through Control Regularization

To address the aforementioned issue, I propose a policy gradient algorithm, CORE-RL (COntrol REgularized Reinforcement Learning), that utilizes a functional regularizer around a, typically suboptimal, control prior (i.e. a controller designed from any prior knowledge of the system). It is shown below that this approach significantly lowers variance in the policy updates, greatly improves sample efficiency of learning, and leads to higher performance policies when compared to both the baseline RL algorithm and the control prior. In addition, this policy is proven to maintain control-theoretic stability guarantees of the control prior throughout the learning process.

Theories and procedures exist to design stable controllers for the vast majority of real-world physical systems (from humanoid robots to robotic grasping to smart power grids). However, conventional controllers for complex systems can be highly suboptimal and/or require great effort in system modeling and controller design.

It would be ideal then to leverage *simple*, suboptimal controllers in RL to reliably learn high-performance policies, which is the focus of this chapter.

Before proceeding, recall the distinction between *controllers* $u : S \rightarrow A$, which are deterministic mappings from state to action, and *policies* $\pi : S \times A \rightarrow [0, 1]$, which are stochastic mappings from which are controllers are drawn. This distinction will be re-emphasized throughout this chapter.

Suppose we have access to a (suboptimal) control prior, $u_{prior} : S \rightarrow A$, and want to combine an RL policy, π_{θ_k}, with this control prior at each learning stage, k. Let us define $u_{\theta_k} : S \rightarrow A$ to represent the realized controller sampled from the stochastic RL policy $\pi_{\theta_k}(a|s)$. I propose to combine the RL policy with the control prior as follows,

$$u_k(s) = \frac{1}{1+\lambda} u_{\theta_k}(s) + \frac{\lambda}{1+\lambda} u_{prior}(s), \tag{3.3}$$

where the action space, A, is assumed to be convex. Note that $u_k(s)$ is the realized controller sampled from stochastic policy π_k, whose distribution over actions has been shifted by u_{prior} such that $\pi_k\left(\frac{1}{1+\lambda} a + \frac{\lambda}{1+\lambda} u_{prior} \,\middle|\, s\right) = \pi_{\theta_k}(a|s)$. The controller u_k in Equation (3.3) is said to be the *mixed controller*, and u_{θ_k} is termed the *RL controller* (drawn from π_k and π_{θ_k}, respectively).

Utilizing the mixed controller (3.3) is equivalent to placing a functional regularizer u_{prior} on the RL controller, u_{θ_k}, with regularizer weight λ. Let $\pi_{\theta_k}(a|s)$ be Gaussian distributed: $\pi_{\theta_k} = \mathcal{N}(\overline{u}_{\theta_k}, \Sigma)$, where Σ describes the user-induced exploration noise, resulting in following expression,

$$\overline{u}_k(s) = \frac{1}{1+\lambda} \overline{u}_{\theta_k}(s) + \frac{\lambda}{1+\lambda} u_{prior}(s). \tag{3.4}$$

The control prior, u_{prior} can be interpreted as a Gaussian prior on the mixed controller, as shown in the following Lemma. Let us define the norm $\|u_1 - u_2\|_\Sigma = (u_1 - u_2)^T \Sigma^{-1} (u_1 - u_2)$.

Lemma 1. *The controller* $\overline{u}_k(s)$ *in Equation* (3.4) *is the solution to the following regularized optimization problem,*

$$\begin{aligned} \overline{u}_k(s) = \arg\min_u \; & \left\| u - \overline{u}_{\theta_k}(s) \right\|_\Sigma \\ & + \lambda ||u - u_{prior}(s)||_\Sigma, \quad \forall s \in S, \end{aligned} \tag{3.5}$$

which can be equivalently expressed as the constrained optimization problem,

$$\begin{aligned} \overline{u}_k(s) = \arg\min_u \; & \left\| u - \overline{u}_{\theta_k}(s) \right\|_\Sigma \\ \text{s.t.} \quad & ||u - u_{prior}(s)||_\Sigma \leq \tilde{\mu}(\lambda) \quad \forall s \in S, \end{aligned} \tag{3.6}$$

where $\tilde{\mu}$ constrains the policy search. Assuming convergence of the RL algorithm, $\overline{u}_k(s)$ converges to the solution,

$$\begin{aligned}\overline{u}_k(s) = \arg\min_{u} \ & \left\| u - \arg\max_{\bar{u}_\theta} \mathbb{E}_{\tau\sim\bar{u}}\Big[r(s,a)\Big]\right\|_{\Sigma} \\ & + \lambda||u - u_{prior}(s)||_{\Sigma}, \quad \forall s \in S \ \text{ as } \ k \to \infty\end{aligned} \tag{3.7}$$

Proof. To prove this Lemma, first equivalence between (3.4) and (3.5) is proven. Then it is shown that convergence of (3.5) to (3.7) directly follows. Finally, equivalence between (3.5) and (3.6) is proven.

Equivalence between (3.4) **and** (3.5) **:** Let $\pi_{\theta_k}(a|s)$ be a Gaussian distributed policy with mean $\overline{u}_{\theta_k}(s)$: $\pi_{\theta_k}(a|s) \sim \mathcal{N}(\overline{u}_{\theta_k}(s), \Sigma)$. Thus, Σ describes exploration noise. From the mixed policy definition (3.4), the following Gaussian distribution is obtained describing the mixed policy:

$$\begin{aligned}\pi_k(a|s) &= \mathcal{N}(\frac{1}{1+\lambda}\overline{u}_{\theta_k} + \frac{1}{1+\lambda}u_{prior}, \Sigma) \\ &= \frac{1}{c_N}\mathcal{N}\Big(\overline{u}_{\theta_k}(s), (1+\lambda)\Sigma\Big) \cdot \mathcal{N}\Big(u_{prior}(s), \frac{1+\lambda}{\lambda}\Sigma\Big),\end{aligned} \tag{3.8}$$

where the second equality follows based on the properties of products of Gaussians. Let us define $\|u_1 - u_2\|_{\Sigma} = (u_1 - u_2)^T\Sigma^{-1}(u_1 - u_2)$, and let $|\Sigma|$ be the determinant of $|\Sigma|$. Then, distribution (3.8) can be rewritten as the product,

$$\begin{aligned}\mathbb{P}(X(s)) = -c_1 \exp(-\frac{1}{2(1+\lambda)}\|X(s) - \overline{u}_{\theta_k}(s)\|_{\Sigma}) \times \\ - c_1\lambda^{\frac{k}{2}} \exp(-\frac{\lambda}{2(1+\lambda)}\|X(s) - u_{prior}(s)\|_{\Sigma}) \\ c_1 = \frac{1}{c_N\sqrt{(2\pi)^k(1+\lambda)^k|\Sigma|}}\end{aligned} \tag{3.9}$$

where $X(s)$ is a random variable with $\mathbb{P}(X(s))$ representing the probability of taking action X from state s under policy (3.4). Further simplifying this PDF, one obtains:

$$\begin{aligned}\mathbb{P}(X(s)) = c_2 \exp\Big(- \|X(s) - \overline{u}_{\theta_k}(s)\|_{\Sigma} \\ - \lambda\|X(s) - u_{prior}(s)\|_{\Sigma}\Big) \\ c_2 = \frac{\lambda^{\frac{k}{2}}}{c_N(2\pi)^k(1+\lambda)^k|\Sigma|}\end{aligned} \tag{3.10}$$

Since the probability $\mathbb{P}(X(s))$ is maximized when the argument of the exponential in Equation (3.10) is minimized, then the maximum probability policy can be expressed

as the solution to the following regularized optimization problem,

$$\begin{aligned}\overline{u}_k(s) = \arg\min_u(s) \;\; & \|u(s) - \overline{u}_{\theta_k}(s)\|_\Sigma + \\ & \lambda\|u(s) - u_{prior}(s)\|_\Sigma, \quad \forall s \in S.\end{aligned} \tag{3.11}$$

Therefore the mixed policy $\overline{u}_k(s)$ from Equation (3.4) is the solution to Problem (3.5).

Convergence of (3.5) **to** (3.7)**:** Note that $\overline{u}_{\theta_k}$ and π_{θ_k} are parameterized by the same θ_k and represent the iterative solution to the optimization problem $\arg\max_\theta \mathbb{E}_{\tau\sim u_k}\Big[r(\tau)\Big]$ at the latest policy iteration. Thus, assuming convergence of the RL algorithm, problem (3.11) can be rewritten as follows,

$$\begin{aligned}\overline{u}_k = \arg\min_u \;\; & \Big\|u(s) - \arg\max_{u_{\theta_k}} \mathbb{E}_{\tau\sim u_k}\Big[r(s,a)\Big]\Big\|^2 \\ & + \lambda||u(s) - u_{prior}(s)||^2, \quad \forall s \in S.\end{aligned} \tag{3.12}$$

Equivalence between (3.5) **and** (3.6) **:** Finally, let us show that the solutions for regularized problem (3.5) and the constrained optimization problem (3.6) are equivalent.

First, note that Problem (3.5) is the dual to Problem (3.6), where λ is the dual variable. Clearly problem (3.5) is convex in u. Furthermore, Slater's condition holds, since there is always a feasible point (e.g. trivially $u(s) = u_{prior}(s)$). Therefore strong duality holds. This means that $\exists\lambda \geq 0$ such that the solution to Problem (3.6) must also be optimal for Problem (3.5).

To show the other direction, fix $\lambda > 0$ and define $R(u) = \|u(s) - \overline{u}_{\theta_k}(s)\|^2$ and $C(u) = ||u(s) - u_{prior}(s)||^2$ for all $s \in S$. Let us denote u^* as the optimal solution for Problem (3.5) with $C(u^*) = \tau > \tilde{\mu}$ (note that $\tilde{\mu}$ can be chosen). However, suppose u^* is *not* optimal for Problem (3.6). Then there exists $\tilde{u}$ such that $R(u^*) < R(\tilde{u})$ and $C(\tilde{u}) \leq \tilde{\mu}$. Denote the difference in the two rewards by $R(\tilde{u}) - R(u^*) = R_{diff}$. Thus the following relations hold,

$$R(\tilde{u}) + \lambda C(\tilde{u}) < R(u^*) + \lambda C(u^*) + R_{diff} + \lambda\Big[\tilde{\mu} - \tau\Big]. \tag{3.13}$$

This leads to the conditional statement,

$$\begin{aligned}& R_{diff} + \lambda\Big[\tilde{\mu} - \tau\Big] \geq 0 \\ & \quad\Rightarrow \quad R(\tilde{u}) + \lambda C(\tilde{u}) < R(u^*) + \lambda C(u^*).\end{aligned} \tag{3.14}$$

For fixed λ, there always exists $\tilde{\mu} > 0$ such that the condition $R_{diff} + \lambda\left[\tilde{\mu} - \tau\right] \geq 0$ holds. However, this leads to a contradiction, since it was assumed that u^* is optimal for Problem (3.5). One can then conclude that $\exists\tilde{\mu}$ such that the solution to Problem (3.5) must be optimal for Problem (3.6). Therefore, Problems (3.5) and (3.6) have equivalent solutions. □

The equivalence between (3.4) and (3.5) illustrates that the control prior acts as a functional regularization (recall that $\overline{u}_{\theta_k}$ solves the reward maximization problem appearing in (3.7)). The policy mixing (3.4) can *also* be interpreted as constraining policy search near the control prior, as shown by (3.6). More weight on the control prior (higher λ) constrains the policy search more heavily. In certain settings, the problem can be solved in the constrained optimization formulation [105].

3.1.1 CORE-RL Algorithm

The learning algorithm is described in Algorithm 1. At the high level, the process can be described as:

- First compute the control prior based on prior knowledge (Line 1). See Section 5 for a controller synthesis example.
- For a given policy iteration, compute the regularization weight, λ, using the strategy described in Section 3.1.3 (Lines 7-9). The algorithm can also use a fixed regularization weight, λ (Lines 10-11).
- Deploy the mixed controller (3.3) on the system, and record the resulting states/action/rewards (Lines 13-15).
- At the end of each policy iteration, update the policy based on the recorded state/action/rewards (Lines 16-18).

3.1.2 Bias-Variance Tradeoff

Theorem 1 below formally states that mixing the policy gradient-based controller, π_{θ_k}, with the control prior, u_{prior}, decreases learning variability. However, the mixing may introduce bias into the learned policy that depends on the (a) regularization λ, and (b) sub-optimality of the control prior. Bias is defined in (3.15) and refers to the difference between the mixed policy and the (potentially locally) optimal RL policy *at convergence*.

Theorem 1. *Consider the mixed policy* (3.3) *where π_{θ_k} is a policy gradient-based RL policy, and denote the (potentially local) optimal policy to be π_{opt}. The variance*

Algorithm 1 Control Regularized RL (CORE-RL)

Compute the control prior, u_{prior} using the known model $f^{known}(s, a)$ (or other prior knowledge)
Initialize RL policy π_{θ_0}
Initialize array $\mathcal{D}$ for storing rollout data
Set $k = 1$ (representing k^{th} policy iteration)
while $k <$ Episodes **do**
    Evaluate policy $\pi_{\theta_{k-1}}$ at each timestep
    if Using Adaptive Mixing Strategy **then**
        At each timestep, compute regularization weight λ
        for the control prior using the TD-error from (3.23).
    else
        Set constant regularization weight λ
    end if
    Deploy mixed policy π_{k-1} from (3.3) to obtain
        rollout of state-action-reward for T timesteps.
    Store resulting data (s_t, a_t, r_t, s_{t+1}) in array $\mathcal{D}$.
    Using data in $\mathcal{D}$, update policy using any policy
        gradient-based RL algorithm (e.g. DDPG, PPO)
        to obtain θ_k.
    $k = k + 1$
end while
return Policy π_{θ_k}, u_{prior} ▷ Overall controller

(3.2) *of the mixed policy arising from the policy gradient is reduced by a factor* $(\frac{1}{1+\lambda})^2$ *when compared to the RL policy with no control prior.*

However, the mixed policy may introduce bias proportional to the sub-optimality of the control prior. Letting $D_{sub} = D_{TV}(\pi_{opt}, \pi_{prior})$*, then the policy bias (i.e.* $D_{TV}(\pi_k, \pi_{opt})$*) is bounded as follows,*

$$\begin{aligned} D_{TV}(\pi_k, \pi_{opt}) &\geq D_{sub} - \frac{1}{1+\lambda} D_{TV}(\pi_{\theta_k}, \pi_{prior}) \\ D_{TV}(\pi_k, \pi_{opt}) &\leq \frac{\lambda}{1+\lambda} D_{sub} \qquad \text{as } k \to \infty \end{aligned} \tag{3.15}$$

where $D_{TV}(\cdot, \cdot)$ *represents the total variation distance between two probability measures (i.e. policies). Thus, if* D_{sub} *and* λ *are large, this will introduce policy bias.*

Proof. Let us define the stochastic action (i.e. random variable) $\mathcal{A}_{k+1}^{act} \sim \pi_{\theta_{k+1}}(a|s)$. Then recall from Equation (3.2) that assuming a fixed, Gaussian distributed policy,

$\pi_{\theta_k}(a|s)$,

$$\text{var}[\mathcal{A}^{act}_{k+1}|s] \approx \alpha^2 \frac{d\pi_{\theta_k}}{d\theta} \text{var}[\nabla_\theta J(\theta_k)] \frac{d\pi_{\theta_k}}{d\theta}^T. \tag{3.16}$$

Based on the mixed policy definition (3.3), the following relation between the variance of π_k and π_{θ_k} (the mixed policy and RL policy, respectively) is obtained,

$$\begin{aligned} \text{var}[\pi_{k+1}] &= \text{var}\Big[\frac{1}{1+\lambda}\mathcal{A}^{act}_{k+1} + \frac{\lambda}{1+\lambda}u_{prior}|s\Big] \\ &= \frac{1}{(1+\lambda)^2}\text{var}[\mathcal{A}^{act}_{k+1}|s] \\ &= \frac{\alpha^2}{(1+\lambda)^2}\frac{d\pi_{\theta_k}}{d\theta}\text{var}[\nabla_\theta J(\theta_k)]\frac{d\pi_{\theta_k}}{d\theta}^T. \end{aligned} \tag{3.17}$$

Compared to the variance (3.2), a variance reduction is observed when utilizing the same learning rate α. Taking the same policy gradient from (3.2), $\text{var}[\nabla_\theta J(\theta_k)]$, then the variance is reduced by a factor of $(\frac{1}{1+\lambda})^2$ by introducing policy mixing.

Lower variance comes at a price – potential introduction of bias into policy. Let us define the policy bias as $D_{TV}(\pi_k, \pi_{opt})$, and let us denote $D_{sub} = D_{TV}(\pi_{opt}, \pi_{prior})$. Since total variational distance, D_{TV} is a metric, the triangle inequality can be used to obtain:

$$D_{TV}(\pi_k, \pi_{opt}) \geq D_{TV}(\pi_{prior}, \pi_{opt}) - D_{TV}(\pi_{prior}, \pi_k). \tag{3.18}$$

The term $D_{TV}(\pi_{prior}, \pi_k)$ can be further broken down:

$$\begin{aligned} &D_{TV}(\pi_{prior}, \pi_k) \\ &\quad= \sup_{(s,a)\in S\text{x}A} \Big|\pi_{prior} - \frac{1}{1+\lambda}\pi_{\theta_k} - \frac{\lambda}{1+\lambda}\pi_{prior}\Big| \\ &\quad= \frac{1}{1+\lambda}\sup_{(s,a)\in S\text{x}A} |\pi_{\theta_k} - \pi_{prior}| \\ &\quad= \frac{1}{1+\lambda}D_{TV}(\pi_{\theta_k}, \pi_{prior}). \end{aligned} \tag{3.19}$$

This holds for all $k \in \mathbb{N}$. Utilizing (3.18) and (3.19) results in the lower bound in (3.15), which is restated below,

$$D_{TV}(\pi_k, \pi_{opt}) \geq D_{sub} - \frac{1}{1+\lambda}D_{TV}(\pi_{\theta_k}, \pi_{prior})$$

To obtain the upper bound, let the policy gradient algorithm with *no* control prior achieve asymptotic convergence to the (locally) optimal policy π_{opt} (as proven for

certain classes of function approximators in [159]). Denote this policy as $\pi_{\theta_k}^{(p)}$, such that $\pi_{\theta_k}^{(p)} \rightarrow \pi_{opt}$ as $k \rightarrow \infty$. In this case, the total variation distance between the mixed policy (3.3) and the optimal policy can be computed as follows,

$$\begin{aligned} & D_{TV}(\pi_{opt}, \pi_k^{(p)}) \\ & \quad = \sup_{(s,a) \in S \text{x} A} |\pi_{opt} - \frac{1}{1+\lambda}\pi_{\theta_k}^{(p)} - \frac{\lambda}{1+\lambda}\pi_{prior}| \\ & \quad = \frac{\lambda}{1+\lambda} \sup_{(s,a) \in S \text{x} A} |\pi_{opt} - \pi_{prior}| \quad \text{as } k \rightarrow \infty \\ & \quad = \frac{\lambda}{1+\lambda} D_{TV}(\pi_{opt}, \pi_{prior}) \quad \text{as } k \rightarrow \infty \\ & \quad = \frac{\lambda}{1+\lambda} D_{sub} \quad \text{as } k \rightarrow \infty. \end{aligned} \tag{3.20}$$

Note that this represents an *upper bound* on the bias, since it assumes that $\pi_{\theta_k}^{(p)}$ is uninfluenced by π_{prior} during learning. It shows that $\pi_{\theta_k}^{(p)}$ is a feasible policy, but not necessarily optimal when accounting for regularization with π_{prior}. This results in the following upper bound, completing the proof:

$$\begin{aligned} D_{TV}(\pi_{opt}, \pi_k) & \leq D_{TV}(\pi_{opt}, \pi_k^{(p)}) \\ & = \frac{\lambda}{1+\lambda} D_{sub} \quad \text{as } k \rightarrow \infty. \end{aligned} \tag{3.21}$$

□

Recall that π_{prior} is the stochastic analogue to the deterministic control prior u_{prior}, such that $\pi_{prior}(a|s) = \mathbb{1}(a = u_{prior}(s))$ where $\mathbb{1}(x)$ is the indicator function returning 1 if x is true. Note that the bias/variance results apply to the *policy* – not the accumulated reward.

Intuition: Figure 3.2 is used to provide some intuition for the control regularization discussed above. Note the following:

1) The explorable region of the state space is denoted by the set $\mathcal{S}_{st}$, which grows as λ decreases and vice versa. This illustrates the constrained policy search interpretation of regularization in the state space.
2) The difference between the control prior trajectory and optimal trajectory (i.e. D_{sub}) *may* bias the final policy depending on the explorable region $\mathcal{S}_{st}$ (i.e. λ). Figure 3.2 shows this difference, and its implications, in state space.
3) If the optimal trajectory is within the explorable region, then it is possible to learn the corresponding optimal policy – otherwise the policy will remain suboptimal.

Points 1 and 3 will be further addressed in Section 3.2.

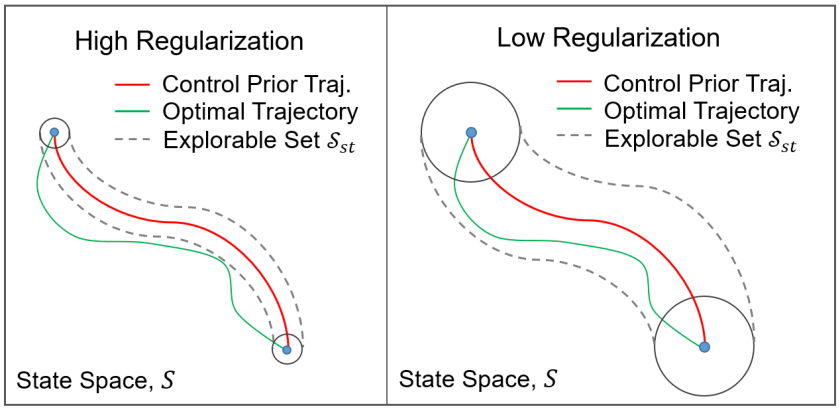


Figure 3.2: Illustration of optimal trajectory vs. control-theoretic trajectory with the explorable set $\mathcal{S}_{st}$. *(a)* With high regularization, set $\mathcal{S}_{st}$ is small, making it impossible to learn the optimal trajectory. *(b)* With lower regularization, set $\mathcal{S}_{st}$ is larger so it *is* possible to learn the optimal trajectory. However, this also enlarges the policy search space.

3.1.3 Computing the mixing parameter λ

A remaining challenge is the automatic tuning of parameter λ, especially as more training data is acquired. While setting a fixed λ can perform well, intuitively, λ should be large when the RL controller is highly uncertain, and it should decrease with higher confidence in the learned controller.

In the multiple model adaptive control (MMAC) framework, a set of controllers (each based on a different underlying model) is generated. A meta-controller computes the overall controller by selecting the weighting for different candidate controllers, based on how close the underlying system model for each candidate controller is deemed to be to the "true" model [101]. Inspired by this approach, the RL controller is weighted proportional to the confidence in its model. This confidence should be state-dependent (i.e. low confidence in areas of the state space where little data has been collected). However, since the RL controller does not utilize a dynamical system model, confidence in the RL controller is measured via the magnitude of the *temporal difference* (TD) error,

$$|\delta^{\pi}(s_t)| = |r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) - Q^{\pi}(s_t, a_t)|, \tag{3.22}$$

where $a_t \sim \pi(a|s_t)$, $a_{t+1} \sim \pi(a|s_{t+1})$. This TD error measures how poorly the RL algorithm predicts the value of subsequent actions from a given state. A high TD-error implies that the estimate of the action-value function at a given state is poor, so the controller should rely more heavily on the control prior (a high λ value). In order to scale the TD-error to a value in the interval $\lambda \in [0, \lambda_{max}]$, the negative

exponential of the TD-error, computed at run-time, is chosen as a useful value of λ

$$\lambda(s_t) = \lambda_{max}\Big(1 - e^{-C|\delta(s_{t-1})|}\Big). \tag{3.23}$$

The parameters C and λ_{max} are tuning parameters of the adaptive weighting strategy. Note that Equation (3.23) uses $\delta(s_{t-1})$ rather than $\delta(s_t)$, because computing $\delta(s_t)$ requires measurement of state s_{t+1}. Thus the method relies on the reasonable assumption that $\delta(s_t) \approx \delta(s_{t-1})$, since s_t should be very close to s_{t-1} in practice.

Equation (3.23) yields a low value of λ if the RL action-value function predictions are accurate. This measure is chosen because the (explicit) underlying model of the RL controller is the value function (rather than a dynamical system model). The experiments presented below show that this adaptive scheme based on the TD error allows better tuning of the variance and performance of the policy.

3.2 Control Theoretic Stability Guarantees

In many controls applications, it is crucial to ensure dynamic stability, not just high rewards, during learning. This is particularly important if the reward function used by the learning algorithm is misspecified. When a (crude) dynamical system model is available, classic controller synthesis tools (i.e. LQR, PID, etc.) can be utilized to obtain a stable control prior in a region of the state space. This section utilizes a well-established tool from robust linear control theory ($\mathcal{H}^\infty$ control), to analyze system stability under the mixed policy (3.3), and prove stability guarantees throughout learning when using a robust control prior.

The results in this chapter are built upon the idea that the control prior should *maximize robustness to disturbances and model uncertainty*. Hence, the RL controller, u_{θ_k}, can be treated as a performance-maximizing "disturbance" to the control prior, u_{prior}. The mixed policy then takes advantage of the stability properties of the robust control prior, *and* the performance optimization properties of the RL algorithm. Principles from $\mathcal{H}^\infty$ control [49] can yield a robust control prior.

Recall that in the MDP model (2.1), the system dynamics were described by the mapping $f^{(a)} : S \times A \to S$, which is unknown to the learning agent. However, suppose that the model can be decomposed into a nominal known part and an unknown part. Then the system evolution can be represented by the following dynamical system and its continuous-time analogue,

$$\begin{aligned} s_{t+1} &= f^{(a)}(s_t, a_t) = f^{known}(s_t, a_t) + f^{unknown}(s_t, a_t), \\ \dot{s} &= f_c^{(a)}(s, a) = f_c^{known}(s, a) + f_c^{unknown}(s, a), \end{aligned} \tag{3.24}$$

where f^{known} captures the nominal dynamics model, $f^{unknown}$ represents the system unknowns, $\dot{s}$ denotes the continuous time-derivative of the state s, and $f_c(s, a)$ denotes the continuous-time analogue of the discrete time dynamics $f(s_t, a_t)$. A control prior can typically be designed from the known part of the system model, f^{known}.

For the nonlinear dynamical system (3.24), linearize the *known* part of the model $f_c^{known}(s, a)$ around a desired equilibrium point to obtain the following,

$$\begin{aligned} \dot{s} &= As + B_1 w + B_2 a \\ z &= C_1 s + D_{11} w + D_{12} a, \end{aligned} \tag{3.25}$$

where $w \in \mathbb{R}^{m_1}$ is the disturbance vector, and $z \in \mathbb{R}^{p_1}$ is the controlled output. This analysis focuses on the continuous-time dynamics rather than discrete-time equivalents, since all mechanical systems have continuous time dynamics that can be discovered through analysis of the system Lagrangian. However, similar analysis can be done for discrete-time dynamics. The following standard assumption – conditions for its satisfaction can be found in [48]–is assumed to hold.

Assumption 1. *A $\mathcal{H}^\infty$ controller exists for linear system* (3.25) *that stabilizes the system in a region of the state space.*

Stability here means that system trajectories are bounded around the origin or a setpoint. An H^∞ controller $u^{\mathcal{H}^\infty}(s) = -Ks$, can be synthesized using established techniques described in [48]. The resulting controller is robust with *worst-case disturbances* attenuated by a factor ζ_k before entering the output, where $\zeta_k < 1$ is a parameter returned by the synthesis algorithm. See Appendix B for further details on $\mathcal{H}^\infty$ control and its robustness properties.

Having synthesized a robust $\mathcal{H}^\infty$ controller for the *linear system model* (3.25), let us now focus on how those robustness properties (e.g. disturbance attenuation by ζ_k) influence the uncertain nonlinear system (3.24) controlled by the mixed policy (3.3). The system dynamics (3.24) can be expressed in terms of the linearization (3.25) plus a disturbance term as follows,

$$\dot{s} = f_c(s, a) = As + B_2 a + d(s, a), \tag{3.26}$$

where $d(s, a)$ gathers together all dynamic uncertainties and nonlinearities. Feedback linearization based on the nominal nonlinear model (3.24) can be used to bound this uncertainty to a smaller value.

The stability of the nonlinear system (3.26) under the mixed policy (3.3) can now be analyzed using Lyapunov theory [96]. Consider the Lyapunov function $V(s) = s^T P s$, where a value of P is obtained during the synthesis of the $\mathcal{H}^\infty$ controller (see Appendix B). If a closed region, $\mathcal{S}_{st}$, can be defined around the origin, such that $\dot{V}(s) < 0$ outside that region, then by standard Lyapunov analysis, $\mathcal{S}_{st}$ is forward invariant and asymptotically stable (note $\dot{V}(s)$ is the time-derivative of the Lyapunov function). Since the $\mathcal{H}^\infty$ control law satisfies an Algebraic Riccati Equation, one obtains the following relation,

Lemma 2. *For any state s, satisfaction of the condition,*

$$2s^T P\Big(d(s,a) + \frac{1}{1+\lambda} B_2 u_e\Big) < s^T (C_1^T C_1 + \frac{1}{\zeta_k^2} P B_1 B_1^T P) s, \tag{3.27}$$

implies that $\dot{V}(s) < 0$.

Proof. Recall the Lyapunov function $V(s) = s^T P s$, where P is taken from the Algebraic Riccati Equation,

$$A^T P + PA + C_1^T C_1 + \frac{1}{\gamma_k^2} P B_1 B_1^T P - P B_2 B_2^T P = 0. \tag{3.28}$$

Take the time derivative of the Lyapunov function as follows:

$$\begin{aligned}
\dot{V}(s) &= \frac{dV}{ds}\dot{s} = 2s^T P\Big(As + B_2 a + d(s,a)\Big) \\
&= s^T(-C_1^T C_1 - \frac{1}{\gamma_k^2} P B_1 B_1^T P) s + 2s^T P d(s,a) + \frac{2}{1+\lambda} s^T P B_2 (u_{\theta_k} - u_{prior}) \\
&= s^T(-C_1^T C_1 - \frac{1}{\gamma_k^2} P B_1 B_1^T P) s + 2s^T P\Big(d(s,a) + \frac{1}{1+\lambda} B_2 u_e\Big).
\end{aligned} \tag{3.29}$$

The second equality comes from the Algebraic Riccati Equation (3.46), which the dynamics satisfy by design of the $\mathcal{H}^\infty$ controller. From these results, it follows directly that if,

$$2s^T P\Big(d(s,a) + \frac{1}{1+\lambda} B_2 u_e\Big) < s^T (C_1^T C_1 + \frac{1}{\gamma_k^2} P B_1 B_1^T P) s,$$

then $\dot{V}(s) < 0$. □

Note that $u_e = u_\theta - u^{\mathcal{H}^\infty}$ denotes the difference between the RL controller and control prior, and (A, B_1, B_2, C_1) come from (3.25). Let us bound the RL control

output such that $\|u_e\|_2 \leq C_\pi$, and define the set $C = \{(s,u) \in (S,A) \;\Big|\; \|u_e\|_2 \leq C_\pi,\ H^\infty$ control is stabilizing$\}$. It is also possible to bound the "disturbance" $\|d(s,a)\|_2 \leq C_D$, for all $s \in C$, and define the minimum singular value $\sigma_m(\zeta_k) = \sigma_{min}(C_1^T C_1 + \frac{1}{\zeta_k^2} P B_1 B_1^T P)$, which reflects the robustness of the control prior (i.e. larger σ_m imply greater robustness). Then using Lemma 2 and Lyapunov analysis, one can derive a conservative set that is guaranteed asymptotically stable and forward invariant under the mixed policy, as described in the following theorem.

Theorem 2. *Assume a stabilizing H^∞ control prior within the set C for the dynamical system* (3.26)*. Then asymptotic stability and forward invariance of the set $\mathcal{S}_{st} \subseteq C$*

$$\mathcal{S}_{st} : \{s \in \mathbb{R}^n : \|s\|_2 \leq \frac{1}{\sigma_m(\zeta_k)} \Big(2\|P\|_2 C_D + \frac{2}{1+\lambda} \|PB_2\|_2 C_\pi \Big),\ s \in C\}, \quad (3.30)$$

is guaranteed under the mixed policy (3.3) *for all $s \in C$. The set $\mathcal{S}_{st}$ contracts as (a) robustness of the control prior is increased (by increasing $\sigma_m(\zeta_k)$), (b) dynamic uncertainty/nonlinearity C_D is decreased, or (c) the weighting λ on the control prior is increased.*

Proof. The proof is divided into two steps,

Step (1): Find a set in which Lemma 2 is satisfied.

Consider the condition in Lemma 2. Since the right-hand side is positive (quadratic), a bound on the stability condition can be computed as follows,

$$|2s^T P d(s,a) + \frac{2}{1+\lambda} s^T P B_2 u_e| < s^T (C_1^T C_1 + \frac{1}{\gamma_k^2} P B_1 B_1^T P) s. \quad (3.31)$$

Clearly any set of s that satisfy condition (3.31) also satisfy the condition in Lemma 2. To find such a set, the terms in Condition (3.31) are bounded as follows,

$$\begin{aligned} &|2s^T P d(s,a) + \frac{2}{1+\lambda} s^T P B_2 u_e| \\ &\qquad \leq \ 2|s^T P d(s,a)| + \frac{2}{1+\lambda} |s^T P B_2 u_e| \\ &\qquad \leq \ 2\|s\|_2 \|P\|_2 C_D + \frac{2}{1+\lambda} \|s\|_2 \|PB_2\|_2 C_\pi, \end{aligned} \quad (3.32)$$

where the first inequality follows from the triangle inequality; the second inequality uses the bounds on the disturbance, C_D, and control input difference C_π, as well as the Cauchy-Schwarz inequality. Now consider the right-hand side of Condition

(3.31). Recall that $\sigma_m(\gamma_k) = \sigma_{min}(C_1^T C_1 + \frac{1}{\gamma_k^2} P B_1 B_1^T P)$, the minimum singular value. Then the following holds,

$$\sigma_m(\gamma_k)\|s\|_2^2 \leq s^T(C_1^T C_1 + \frac{1}{\gamma_k^2} P B_1 B_1^T P)s \tag{3.33}$$

Using the bounds in (3.32) and (3.33), Condition (3.31) is guaranteed to be satisfied if the following holds,

$$2\|s\|_2\|P\|_2 C_D + \frac{2}{1+\lambda}\|s\|_2\|PB_2\|_2 C_\pi < \sigma_m(\gamma_k)\|s\|_2^2 \tag{3.34}$$

The set for which condition (3.34) is satisfied can be described by,

$$C \setminus \mathcal{S}_{st} : \{s \in \mathbb{R}^n : \|s\|_2 > \frac{1}{\sigma_m(\gamma_k)}\Big(2\|P\|_2 C_D + \frac{2}{1+\lambda}\|PB_2\|_2 C_\pi\Big),\ s \in C\}. \tag{3.35}$$

Recall that C is the set wherein $\mathcal{H}^\infty$ is a stabilizing controller. From Lemma 2, $\dot{V}(s) < 0$ for all $s \in C \setminus \mathcal{S}_{st}$ described by the set (3.35).

Step (2): Establish stability and forward invariance of $\mathcal{S}_{st}$.

The Lyapunov function $V(s) = s^T P s$ decreases towards the origin, and it was already established that the time derivative of the Lyapunov function is negative for s in set (3.35). Therefore, any state s described by the set (3.35) (intersected with C) must move towards the origin (i.e. towards $\mathcal{S}_{st}$). This follows directly from the properties of Lyapunov functions. Therefore, the set $\mathcal{S}_{st}$ described in (3.30) must be asymptotically stable and forward invariant for all $s \in C$. □

Put simply, Theorem 2 says that all states in C will converge to (and remain within) set $\mathcal{S}_{st}$ under the mixed policy (3.3). Therefore, the stability guarantee is stronger if $\mathcal{S}_{st}$ has smaller cardinality. The set $\mathcal{S}_{st}$ is drawn pictorally in Figure 3.2, and essentially dictates the region that can be explored while maintaining system stability. Note that $\mathcal{S}_{st}$ is *not* the region of attraction.

Theorem 2 highlights the tradeoff between the robustness parameter, ζ_k, of the control prior, the nonlinear uncertainty in the dynamics, C_D, and the utilization of the learned controller, λ. If a more robust control prior (higher $\sigma_m(\zeta_k)$) is available or one has better knowledge of the dynamics (smaller C_D), the learned controller can be heavily weighted (lower λ) during the learning process, while still guaranteeing stability.

While shrinking the set $\mathcal{S}_{st}$ and achieving asymptotic stability along a trajectory or equilibrium point may seem desirable, Figure 3.2 illustrates why this is not

necessarily the case in a reinforcement learning context. The optimal trajectory for a desired task will likely deviate from the nominal trajectory (i.e. the control theoretic-trajectory), as shown in Figure 3.2 – the set $\mathcal{S}_{st}$ illustrates the explorable region under regularization. Figure 3.2(a) shows that strict stability of the nominal trajectory is not always desired, and instead *limited* flexibility (a sufficiently large $\mathcal{S}_{st}$) to explore is preferred. Increasing the weighting on the learned policy π_{θ_k} (decreasing λ) expands the set $\mathcal{S}_{st}$ and allows for greater exploration around the nominal trajectory (at the cost of stability), as seen in Figure 3.2(b).

Remark 5. This section does *not* advocate for $\mathcal{H}^\infty$ control as a general-purpose method for controller prior synthesis in RL algorithms; indeed, the set of problems it can be applied to is much smaller than the set of problems that can be tackled with reinforcement learning. The main message is that in the synthesis of a control prior, *robustness* should be prioritized over *performance* (e.g. $\mathcal{H}^\infty$ over LQG), since RL is excellent at learning high-performance controllers, but notoriously bad at finding robust ones. This allows the interpretation of the idea proposed in this chapter: the RL policy can be viewed as a performance-maximizing "disturbance."

3.3 Empirical Results

To illustrate the characteristics of the CORE-RL Algorithm, it is applied to three problems: (1) cartpole stabilization, (2) car-following control with experimental data, and (3) racecar driving with the TORCS simulator. The existing algorithms DDPG, PPO or TRPO [109, 147, 148] are used as policy gradient RL algorithms, though any similar RL algorithm could be used. All code can be found at [1].

Note that the results presented below focus on reward rather than bias. Bias (as defined in Section 3.1.2) assumes convergence to a (locally) optimal policy, and does not include many factors influencing performance (e.g. slow learning, failure to converge, etc.). In practice, Deep-RL algorithms often do not converge (or take very long to do so). Therefore, reward better demonstrates the influence of control regularization on performance, which is of greater practical interest.

3.3.1 The CartPole Problem

The CORE-RL algorithm is applied to the control of the cartpole from the OpenAI gym environment (*CartPole-v1*). The CartPole environment was modified so that it takes a *continuous* input, rather than discrete input, and the chosen reward function encourages the cartpole to maintain its $x-$position while keeping the pole upright. Further details on the environment and reward function are in Appendix A. To obtain

a control prior, a crude physical model (i.e. linearization of the nonlinear dynamics with $\approx 60\%$ error in the mass and length values) is assumed, and from this an $\mathcal{H}^\infty$ controller can be synthesized. Using this control prior, Algorithm 1 is deployed with several different regularization weights, λ. For each λ, CORE-RL is run 6 times with different random seeds.

Figures 3.4a, 3.5a, 3.6a plot reward improvement over the control prior, which shows that the regularized controllers perform much better than the baseline RL algorithm (either PPO, TRPO, DDPG) in terms of variance, reward, and learning speed. It can also be seen that intermediate values of λ (i.e. $\lambda \approx 4$) result in the best learning, demonstrating the importance of policy regularization.

Figure 3.4b, 3.5b, 3.6b better illustrate the performance-variance tradeoff. For small λ, high variance *and* poor performance are seen. Intermediate values λ result in higher performance and lower variance. As the value of λ is further increased, variance continues to decrease, but the performance also decreases since policy exploration is heavily constrained. The adaptive mixing strategy performs very well: it exhibits low variance through learning, and converges on a high-performance policy.

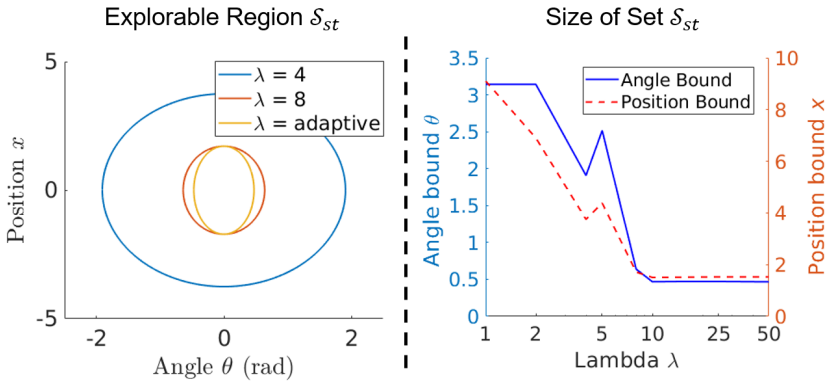


Figure 3.3: Stability region for CartPole under mixed policy. *(a)* Illustration of the stability region for different regularization, λ. For each λ shown, the trajectory goes to and remains within the corresponding stability set throughout training. *(b)* Size of the stability region in terms of the angle θ, and position x. As λ increases, the system trajectory is guaranteed to remain closer to the equilibrium point during learning.

While Lemma 2 proved that the mixed controller (3.4) has the same *optimal* solution as optimization problem (3.5), when computational experiments were tried using the loss in Equation (3.5), the performance (i.e. reward) was found to be worse than CORE-RL and still suffered high variance. In addition, learning with pre-training

on the control prior likewise exhibited high variance and had worse performance than CORE-RL.

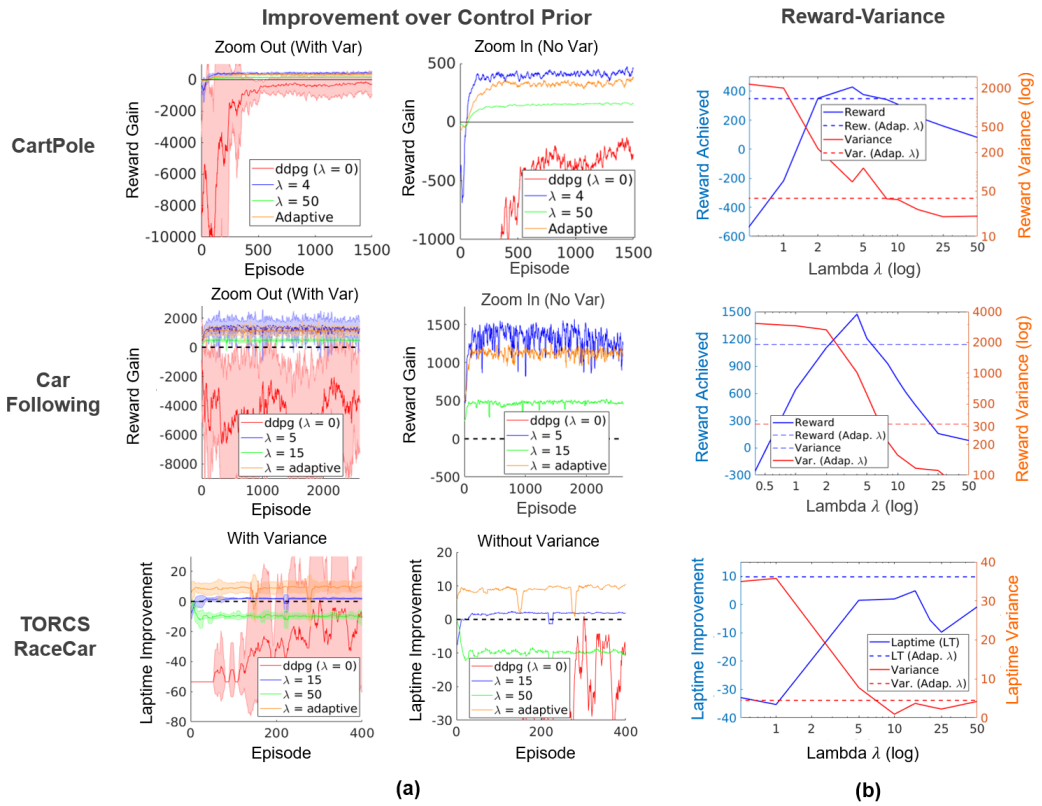


Figure 3.4: Learning results for CartPole, Car-Following, and TORCS RaceCar Problems using DDPG. *(a)* Reward improvement over control prior with different set values for λ or an adaptive λ. The right plot is a zoomed-in version of the left plot without variance bars for clarity. Values above the dashed black line signify improvements over the control prior. *(b)* Performance and variance in the reward as a function of the regularization λ, across different runs of the algorithm using random initializations/seeds. Dashed lines show the performance (i.e. reward) and variance using the adaptive weighting strategy. Variance is measured for all episodes across all runs. Adaptive λ and intermediate values of λ exhibit best learning. Again, performance is baselined to the control prior, so any performance value above 0 denotes improvement over the control prior.

Importantly, according to Theorem 2, the system should maintain stability (i.e. remain within an invariant set around a desired equilibrium point) *throughout the learning process*, and the stable region should shrink as λ is increased. Simulations exhibit exactly this property, as seen in Figure 3.3, which shows the *maximum* deviation from the equilibrium point across all episodes. The system converges to a stability region throughout learning, and this region contracts with increasing λ. Therefore, regularization not only improves learning performance and decreases variance, but can capture stability guarantees from a robust control prior.

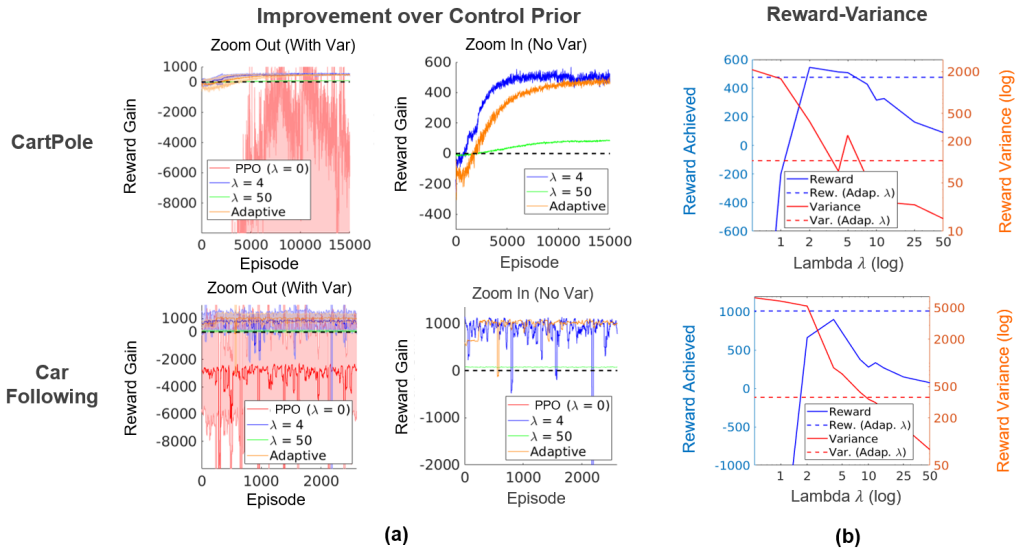


Figure 3.5: Learning results analogous to Figure 3.4, with PPO used as the RL algorithm. TORCS environment excluded because the PPO agent could not complete a lap during the learning stage.

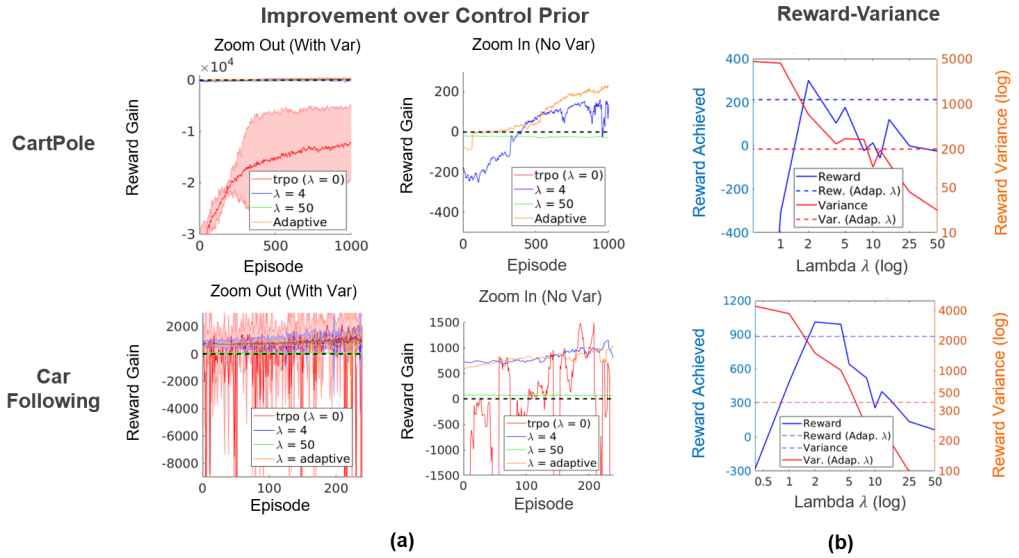


Figure 3.6: Learning results analogous to Figure 3.4, with TRPO used as the RL algorithm. TORCS environment excluded because the TRPO agent could not complete a lap during the learning stage.

3.3.2 Experimental Car-Following

This section applies the CORE-RL framework to experimental data obtained from a chain of 5 cars following each other on an 8-mile segment of a single-lane public road. Position (via GPS), velocity, and acceleration data was recorded from each of the cars, and the acceleration/deceleration of the 4^{th} car in the chain is controlled. The goal is to learn an optimal controller for this car that maximizes fuel efficiency

while avoiding collisions. The experimental setup and data collection process are described in [73]. The control prior is a bang-bang controller that (inefficiently) tries to maintain a large distance from the car in front and behind the controlled car. The reward function penalizes fuel consumption and collisions (or near-collisions). Specifics of the control prior, reward function, and experiments are in Appendix A.

The experimental data is split into 10 second "episodes." The episode data is shuffled, and CORE-RL is applied six times with different random seeds (for several different λ).

Figures 3.4a, 3.5a, 3.6a show again that the regularized controllers perform much better than the baseline RL algorithm for the car-following problem, and demonstrates that regularization leads to performance improvements over the control prior and gains in learning efficiency. Figures 3.4b, 3.5b, 3.6b reinforce that intermediate values of λ (i.e. $\lambda \approx 5$) exhibit optimal performance. Low values of λ exhibit significant deterioration of performance, because the car must learn (with few samples) in a much larger policy search space; the RL algorithm does not have enough data to converge on an optimal policy. High values of λ also exhibit lower performance because they heavily constrain learning. Intermediate λ allow for the best learning using the limited number of experiments.

An adaptive strategy for setting λ (or alternatively tuning to an optimal λ) yields high-performance policies that improve upon both the control prior and RL baseline controller. The variance is also low, so that the learning process *reliably* learns a good controller.

3.3.3 Simulated Driving in the TorcsEnvironment

Finally, CORE-RL is used to generate controllers for cars in *The Open Racing Car Simulator* (Torcs) [180]. The simulator provides readings from 29 sensors, which describe the environment state. The sensors provide information like car speed, distance from track center, wheel spin, etc. The controller decides values for the acceleration, steering, and braking actions taken by the car.

A simple PID-like linearized controller for each action can be used as a control prior for this environment similar to the one described in [168]. These types of controllers are known to have sub-optimal performance, while still being able to drive the car around a lap. All experiments use the CG-Speedway track in Torcs. For each value of λ, the algorithm is applied 5 times with different initializations and random seeds.

For Torcs, *laptime improvement over the control prior* is recorded so that values

above zero denote improved performance over the prior. The laps are timed out at 150s, and the objective is to minimize lap-time by completing a lap as fast as possible. Due to the sparsity of the lap-time signal, a pseudo-reward function was used during training that provides a heuristic estimate of the agent's performance at each time step during the simulation (details in Appendix A).

Once more, Figure 3.4a shows that regularized controllers perform better on average than the baseline RL algorithm, and improves upon the control prior. Figure 3.4b shows that intermediate values of λ exhibit good performance, but using the adaptive strategy for setting λ in the TORCS setting yields the highest-performance policy that significantly beats both the control prior and RL baseline. Also, the variance with the adaptive strategy is significantly lower than for the RL baseline, which again shows that the learning process *reliably* learns a good controller.

Note that PPO or TRPO results are not shown for the TORCS Racecar because the baseline PPO or TRPO learning agents could not complete a lap during the learning process. The code for the PPO, TRPO, and DDPG agent for each environment can be found at [1].

3.4 Conclusion

A significant criticism of RL is that it is unreliable and fragile [90] (i.e. not robust); running multiple learning iterations with random seeds can produce vastly different behaviors, limiting the application of RL to real-world systems. Intuitively, it makes a lot of sense that an algorithm that tries to learn complex behaviors in an extremely large search space will have extreme variability in learned behaviors based on randomness in early exploration. Therefore, this chapter proposed control regularization as a means to intelligently constrain this search space.

Through theoretical results and experimental validation, the proposed method of control regularization substantially alleviates the aforementioned problems, enabling significant variance reduction as well as sample complexity improvements in RL. This method also allows the dynamic stability and robustness properties from a robust control prior to guarantee stability during the learning process, and has the added benefit that it can easily incorporate different RL algorithms (e.g. PPO, DDPG, etc.).

The main limitation of the proposed approach is that it relies on the existence of a control prior. However, in most robotic applications, access to such a control prior is a reasonable assumption. For example, it may be very difficult for a robot arm to

reliably complete a precision insertion task, but a control prior that brings the arm to the general vicinity of the goal can be easily synthesized. Furthermore, with limited expert data, this control prior could even be obtained through imitation learning.

Appendix A: Description of Experiments

Experimental Car-Following

In the original car-following experiments, a chain of 8 cars followed each other on an 8-mile segment of a single-lane public road. Position (via GPS), velocity, and acceleration data were recorded from each of the cars. This data was cut into 4 sets of chains of 5 cars, in order to maximize the data available to learn from. This was further cut into 10 second "episodes" (100 data points each). These training episodes were then randomly shuffled before each run and fed to the algorithm, which learned the controller for the 4^{th} car in the chain.

The reward function used in learning is described below:

$$
\begin{aligned}
r &= -\dot{v}\min(0, a) - 100|G_1(s)| - 50G_2(s), \\
G_1(s) &= \begin{cases} \frac{1}{s_{front} - s_{curr}} & \text{if } s_{front} - s_{curr} \leq 10 \\ \frac{1}{s_{curr} - s_{back}} & \text{if } s_{curr} - s_{back} \leq 10 \\ 0 & \text{otherwise} \end{cases} \\
G_2(s) &= \begin{cases} 1 & \text{if } s_{front} - s_{curr} \leq 2 \\ 1 & \text{if } s_{curr} - s_{back} \leq 2 \\ 0 & \text{otherwise} \end{cases}
\end{aligned} \tag{3.36}
$$

where s_{curr}, s_{front}, and s_{back} denote the position of the controlled car, the car in front of it, and the car behind it. Also, a denotes the control action (i.e. acceleration/deceleration), and $\dot{v}$ denotes the velocity of the controlled car. Therefore, the first term represents the fuel efficiency of the controlled car, and the other terms encourage the car to maintain headway from the other cars and avoid collision.

The control prior utilized is a simple bang-bang controller that (inefficiently) tries to keep us between the car and front and back. It is described by,

$$
\begin{aligned}
a &= \begin{cases} 2.5 & \text{if } K_p \Delta s + K_d \Delta v > 0 \\ -5 & \text{if } K_p \Delta s + K_d \Delta v < 0 \\ 0 & \text{otherwise} \end{cases} \\
\Delta s &= s_{front} - 2s_{curr} - s_{back} \\
\Delta v &= v_{front} - 2v_{curr} - v_{back}
\end{aligned} \tag{3.37}
$$

where v_{curr}, v_{front}, and v_{back} denote the velocity of the controlled car, the car in front of it, and the car behind it. The constants were set as $K_p = 0.4$ and $K_d = 0.5$.

Essentially, the control prior tries to maximize the distance from the car in front and behind, taking into account velocities as well as positions.

TORCS Racecar Simulator

In its full generality TORCS provides a rich environment with input from up to 89 sensors, and optionally the 3D graphic from a chosen camera angle in the race. The controllers have to decide the values of up to 5 parameters during game play, which correspond to the acceleration, brake, clutch, gear and steering of the car. Apart from the immediate challenge of driving the car on the track, controllers also have to make race-level strategy decisions, like making pit-stops for fuel. A lower level of complexity is provided in the *Practice Mode* setting of TORCS. In this mode, all race-level strategies are removed. Currently, state-of-the-art DRL models are capable of racing only in Practice Mode, and this is also the environment considered in this work. In this mode, the input from 29 sensors is considered and used to decide values for the acceleration, steering, and brake actions.

The control prior utilized is a linear controller of the form:

$$K_p(\epsilon - o_i) + K_i \sum_{j=i-N}^{i} (\epsilon - o_j) + K_d(o_{i-1} - o_i), \tag{3.38}$$

where o_i is the most recent observation provided by the simulator for a chosen sensor, and N is a predetermined constant. One controller is used for each of the actions, acceleration, steering, and braking.

The pseudo-reward used during training is given by:

$$r_t = V\cos(\theta) - V\sin(\theta) - V|\texttt{trackPos}|. \tag{3.39}$$

Here, V is the velocity of the car, θ is the angle the car makes with the track axis, and `trackPos` provides the position on the track relative to the track's center. This reward captures the aim of maximizing the longitudinal velocity, minimizing the transverse velocity, and penalizing the agent if it deviates significantly from the center of the track.

CartPole Stabilization

The CartPole simulator is implemented in the OpenAI gym environment ('CartPole-v1'). The dynamics are the same as in the default, as described below,

$$\begin{aligned}
\theta_{t+1} &= x_t + \dot{x}\tau, \\
\dot{\theta}_{t+1} &= \dot{\theta}_t + \Big(\frac{Mg\sin\theta - F\cos\theta - ml\dot{\theta}^2\sin\theta\cos\theta}{\frac{4}{3}Ml - ml\cos^2\theta}\Big)\tau, \\
x_{t+1} &= x_t + \dot{x}\tau, \\
\dot{x}_{t+1} &= \dot{x}_t + \Big(\frac{F + ml\dot{\theta}^2\sin\theta - ml\ddot{\theta}\cos\theta}{M}\Big)\tau,
\end{aligned} \tag{3.40}$$

where the only modification made is that the force on the cart can take on a continuous value, $F \in [-10, 10]$, rather than 2 discrete values, making the action space much larger. Since the control prior can already stabilize the CartPole, the reward is also modified to characterize *how well* the control stabilizes the pendulum. The reward function is stated below, and incentivizes the CartPole to keep the pole upright while minimizing movement in the x-direction:

$$r = -100|\theta| - 2x^2. \tag{3.41}$$

Appendix B: Control Theoretic Stability Guarantees

This section in the appendix provides more background on the material in Section 3.2, going into further detail on the $\mathcal{H}^\infty$ problem definition. Consider the linear dynamical system described by:

$$\begin{aligned}
\dot{s} &= As + B_1 w + B_2 a \\
z &= C_1 s + D_{11} w + D_{12} a \\
y &= C_2 s + D_{21} w + D_{22} a
\end{aligned} \tag{3.42}$$

where $w \in \mathbb{R}^{m_1}$ is the disturbance vector, $u \in \mathbb{R}^{m_1}$ is the control input vector, $z \in \mathbb{R}^{p_1}$ is the error vector (controlled output), $y \in \mathbb{R}^{p_2}$ is the observation vector, and $s \in \mathbb{R}^n$ is the state vector. The system transfer function is denoted,

$$\begin{aligned}
P^s(s) &= \begin{pmatrix} P^s_{11} & P^s_{12} \\ P^s_{21} & P^s_{22} \end{pmatrix} \\
&= \begin{pmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{pmatrix} + \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} (sI - A)^{-1} \begin{bmatrix} B_1 & B_2 \end{bmatrix},
\end{aligned} \tag{3.43}$$

where A, B_i, C_i, D_{ij} are defined by the system model (3.42). Let us make the following assumptions,

- The pairs (A, B_2) and (C_2, A) are stabilizable and observable, respectively.
- The algebraic Riccati equation $A^T P + PA + C_1^T C_1 + P(B_2 B_2^T - \frac{1}{\gamma_k^2} B_1 B_1^T)P = 0$ has positive-semidefinite solution P.
- The algebraic Riccati equation $AP_Y + P_Y A^T + B_1^T B_1 = P_Y(C_2 C_2^T - \frac{1}{\gamma_k^2} C_1 C_1^T)P_Y$ has positive-semidefinite solution P_Y.
- The matrix $\gamma I - P_Y P$ is positive definite.

Under these assumptions, existence of a stabilizing linear $\mathcal{H}^\infty$ controller, $u^{\mathcal{H}^\infty} = -Ks$, is guaranteed [48]. The closed-loop transfer function from disturbance, w, to controlled output, z, is:

$$T_{wz} = P_{11}^s + P_{12}^s K(I - P_{22}^s K)^{-1} P_{21}^s. \tag{3.44}$$

Let $\sigma(\cdot)$ denote the maximum singular value of the argument, and recall that $\|T_{wz}\|_\infty := \sup_w \sigma(T_{wz}(jw))$. Then the H_∞ controller solves the problem,

$$\min_K \sup_w \sigma(T_{wz}(jw)) = \gamma_k, \tag{3.45}$$

to give us controller $u^{\mathcal{H}^\infty} = -Ks$. This generates the maximally robust controller so that the *worst-case disturbance* is attenuated by factor γ_k in the system before entering the controlled output. The H_∞ controller can be synthesized using techniques described in [48].

The H_∞ controller is defined as $u^{\mathcal{H}_\infty} = -B_2^T P x$, where P is a positive symmetric matrix satisfying the Algebraic Riccati equation,

$$A^T P + PA + C_1^T C_1 + \frac{1}{\gamma_k^2} PB_1 B_1^T P - PB_2 B_2^T P = 0, \tag{3.46}$$

where (A, B_1, B_2, C_1) are defined in (3.42). The result is that the control law $u^{\mathcal{H}_\infty}$ stabilizes the system with disturbance attenuation $\|T_{wz}\|_\infty \leq \gamma_k$.

Since the system being considered is nonlinear, one must consider a modification to the dynamics (3.42) that *linearizes* the dynamics about some equilibrium point and gathers together all non-linearities and disturbances,

$$\dot{s} = f_c(s, a) = As + B_2 a + d(s, a), \tag{3.47}$$

where $d(s, a)$ captures dynamic uncertainty/nonlinearity as well as disturbances. To keep this small, one could use feedback linearization based on the nominal nonlinear model (3.24), but this is outside the scope of this work.

Consider the Lyapunov function $V(s) = s^T Ps$, where P is taken from Equation (3.46). Stability of the uncertain system (3.26) can be analyzed under the mixed policy (3.3) using Lyapunov analysis. Lemma 2 is used to compute a set $\mathcal{S}_{st}$ such that $\dot{V}(s) < 0$ in a region outside that set. Satisfaction of this condition guarantees forward invariance of that set [96], as well as its asymptotic stability (from the region for which $\dot{V}(s) < 0$). By bounding terms as described in Section 3.2, one can conservatively compute the set $\mathcal{S}_{st}$ for which $\dot{V}(s) < 0$, which is shown in Theorem 2.

Chapter 4

SAFE REINFORCEMENT LEARNING THROUGH CONTROL BARRIER FUNCTIONS

Even if one were able to guarantee reliable, sample-efficient reinforcement learning (RL), such a guarantee will often not be sufficient for real-world deployment of RL. In many applications, it is important to guarantee, with very high confidence, that the controlled system will be safe (e.g. remain stable, avoid collisions) throughout its operation, including the learning stage. However, since RL focuses predominantly on maximizing long-term reward, it is likely to explore unsafe behaviors during its learning process. This issue will be exacerbated if the reward function is even slightly misspecified. This is problematic for any RL algorithm that will be deployed on hardware, as exploring unsafe policies during the learning or deployment phases could damage the hardware or bring harm to a human. If a robot performs its task flawlessly for 120 hours, but breaks someone's arm every 121^{st} hour, very few people would find such performance acceptable.

The first half of this chapter develops a framework for integrating existing reinforcement learning algorithms with control barrier functions (CBF) to both guarantee safety and improve exploration efficiency in RL, even with uncertain model information. The CBF requires a (potentially poor) nominal dynamics model, but ensures online safety of nonlinear systems during the entire learning process and can help guide exploration of the policy space. An on-line process learns a more accurate model of the governing dynamical system over time. This methodology effectively constrains the policy exploration process to a set of safe polices defined by the CBF.

It is important to note that direct inclusion of a safety shield/filter [10, 66] (e.g. CBF, HJI) into most RL frameworks will lead to a distortion of any RL policy updates, likely leading to poor learning. Therefore, an important issue tackled in this chapter is how to properly integrate safety mechanisms or shields into most reinforcement learning frameworks, while accounting for (and taking advantage of) distortion of the policy updates.

The second half of the chapter (Sections 4.4 and 4.5) extends these results to consider a more general class of multi-agent CBFs with correlated, multivariate uncertainty. This approach enables improved applicability to robots operating in multi-agent,

uncertain environments.

4.1 Enforcing Safety during Learning through CBFs

Recall the MDP defined in Section 2. Our goal is to learn an optimal policy π with respect to that MDP while maintaining safety throughout the learning process under uncertain knowledge of the system dynamics. However, let us now consider control-affine dynamics (a good assumption when dealing with robotic systems), such that the time evolution of the system can be given by

$$s_{t+1} = f^{(n)}(s_t, a_t) = f(s_t) + g(s_t)a_t + d(s_t), \tag{4.1}$$

where $f : S \to \mathbb{R}^n$ is the nominal unactuated dynamics, $g : S \to \mathbb{R}^{n,m}$ is the nominal actuated dynamics, and $d : S \to \mathbb{R}^n$ is the *unknown* system dynamics. Therefore, f and g compose a known nominal model of the dynamics, and d represents the uncertainty. In practice, the nominal model may be quite bad (e.g. a robot model that ignores friction and compliance), and a much better dynamic model must be learned from data.

Given the simplification of these dynamics, the discrete-time control barrier function from Section 2.2 is redefined as follows.

Definition 2. Given a set $C \in \mathbb{R}^n$ defined by (2.7), the smooth function $h : \mathbb{R}^n \to \mathbb{R}$ is a *discrete-time control barrier function* (CBF) for dynamical system (4.1) if there exists $\eta \in [0, 1]$ such that for all $s_t \in C$,

$$\sup_{a \in A} \Big[h\Big(f(s_t) + g(s_t)a + d(s_t) \Big) + (\eta - 1)h(s_t) \Big] \geq 0, \tag{4.2}$$

where the associated control barrier condition (CBC) corresponding to CBF, h, is

$$h\Big(f(s_t) + g(s_t)a + d(s_t) \Big) + (\eta - 1)h(s_t) \geq 0. \tag{4.3}$$

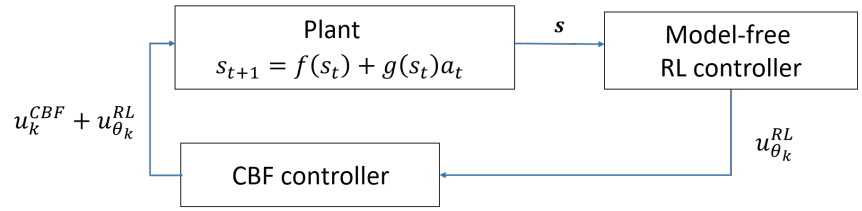
In this section, let us restrict attention to *affine* control barrier functions of form $h = p^T s + q$, ($p \in \mathbb{R}^n, q \in \mathbb{R}$), which will enable more efficient CBF computations. However, this methodology could support more general control barrier functions, and the following section will examine a broader set of *multi-agent* control barrier functions.

As discussed in the introduction, the CBF in Definition 2 can be used to formulate an optimization problem (similar to (2.10)) that implicitly defines a safe controller,

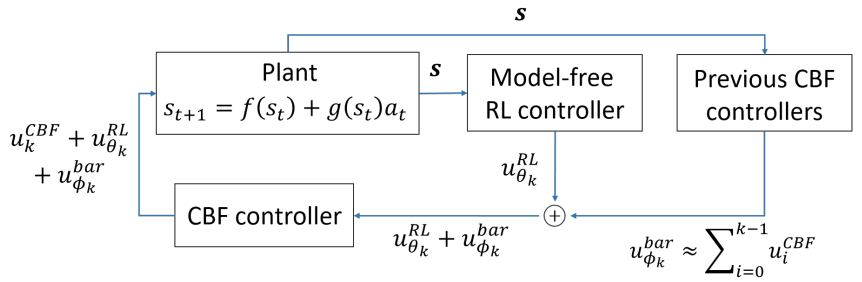
u^{CBF}, which guarantees forward invariance of set C. However, since the goal is to leverage an RL controller, it may be desirable to use the CBF to only *filter* the proposed RL action. Therefore, let us define the following policy/controller,

$$\begin{aligned} u_k(s) &= u^{RL}_{\theta_k}(s) + u^{CBF}_k(s, u^{RL}_{\theta_k}), \\ \pi_k(a|s) &= \pi^{RL}_{\theta_k}(a|s) * \pi^{CBF}_k(a|s, \pi^{RL}_{\theta_k}), \end{aligned} \tag{4.4}$$

where $\pi^{CBF}_k(a|s) = (a = u^{CBF}_k(s))$ with $\mathbb{1}(x)$ being the indicator function and $*$ being the convolution operator. The policy/controller in (4.4) combines a model-free RL policy (parameterized by θ_k) and a CBF-based controller in the architecture shown in Figure 4.1a.



(a)



(b)

Figure 4.1: Control architecture combining model-free RL policy with model-based CBF to guarantee safety. (a) Initial architecture that uses CBF to *compensate* for unsafe control actions, but does not guide learning and exploration. (b) Architecture that uses CBF to guide exploration and learning, as well as ensure safety.

Remark 6. Note that t indexes timesteps *within* each policy iteration or trial, whereas k indexes the policy iterations (which contain trajectories with several time steps). The CBF controller generates control updates throughout the task (computed at each time step, t), whereas the RL policy and GP model update at episodic policy iteration intervals indexed by k.

For the overall controller $u_k(s)$ to be safe, one need only enforce the CBC condition for $a_t = u_k(s_t)$ at each time step, as demonstrated in the following optimization

problem,

$$
\begin{aligned}
(a_t, \epsilon_t) = \underset{a \in A}{\text{argmin}} \quad & \|a\|_2 + K_\epsilon \epsilon \\
\text{s.t.} \quad & p^T f(s_t) + p^T g(s_t)\big(a + u^{RL}_{\theta_k}(s_t)\big) + p^T d(s_t) \\
& \qquad + q \geq (1-\eta) h(s_t) - \epsilon \\
& a^i_{low} \leq a^i + u^{RL,i}_{\theta_k}(s_t) \leq a^i_{high} \ \text{ for } i = 1, ..., m,
\end{aligned}
\tag{4.5}
$$

where ϵ is a slack variable in the safety condition, and K_ϵ is a large constant that penalizes safety violations. The optimization is not sensitive to the K_ϵ parameter as long as it is very large (e.g. 10^{10}), such that safety constraint violations are heavily penalized. Note that this optimization problem is now a quadratic program (QP) as the CBC constraint is linear in the control action, enabling efficient computation of $u^{CBF}(s)$.

However, recall that $d(s)$ is unknown, and therefore must be estimated before it can be included in (4.5). As described in Section 2.3, an updating GP model can be used to directly estimate the posterior distribution over function, $d(s)$, from measurement data. Given a sequence of measurements (s_t, a_t, s_{t+1}) over a horizon T, one can easily compute the uncertain variable, $d_{t-T}, ..., d_{t-1}$ over that horizon using the dynamics model (4.1) (i.e. $d(s_t) = s_{t+1} - f(s_t) - g(s_t)a_t$). Using these measurements, one can directly estimate the mean $\mu_d(s)$ and variance $\sigma^2_d(s)$ at query point s from (2.14). From Equation (2.15), it is known that $|\mu_d(s) - d(s)| \leq k_\delta \sigma_d(s)$ with probability $(1-\delta)$. Therefore, given a desired safety threshold δ, one can bound the CBC utilized in (4.5) while accounting for the uncertainty in $d(s)$. The QP defining u^{CBF} can then be formulated as,

$$
\begin{aligned}
(a_t, \epsilon_t) = \underset{a \in A, \epsilon \in \mathbb{R}_+}{\text{argmin}} \quad & \|a\|_2 + K_\epsilon \epsilon \\
\text{s.t.} \quad & p^T f(s_t) + p^T g(s_t)\big(a + u^{RL}_{\theta_k}(s_t)\big) + p^T \mu_d(s_t) - \\
& \qquad k_\delta |p|^T \sigma_d(s_t) + q \geq (1-\eta) h(s_t) - \epsilon \\
& a^i_{low} \leq a^i + u^{RL,i}_{\theta_k}(s_t) \leq a^i_{high} \ \text{ for } i = 1, ..., m
\end{aligned}
\tag{4.6}
$$

The solution to this optimization problem (4.6) enforces the safety condition (4.3) as best as possible with minimum control effort, even with uncertain dynamics. Accounting for the dynamics uncertainty through GP models allows us to certify system safety, even with a poor nominal model.

Define the set $C_\epsilon : \{s \in \mathbb{R}^n : h(s) \geq -\frac{\epsilon}{\eta}\}$. Then the following lemma can be proved.

Lemma 3. *For dynamical system* (4.1)*, if there exists a solution to* (4.6) *for all* $s \in C$ *with* $\epsilon = 0$*, then the controller derived from* (4.6) *renders set* C *forward invariant with probability* $(1 - \delta)$.

However, suppose instead that there exists $s \in C$ *such that* (4.6) *has solution with* $\epsilon = \epsilon^{max} > 0$*. If for all* $s \in C_\epsilon$*, the solution to* (4.6) *satisfies* $\epsilon \leq \epsilon^{max}$*, then the larger set* C_ϵ *is forward invariant with probability* $(1 - \delta)$.

Proof. The first part of the lemma follows directly from Definition 2 and the probabilistic bounds on the uncertainty obtained from GPs shown in equation (2.15).

For the second part, the property of GPs in equation (2.15) implies that with probability $(1 - \delta)$, the following inequality is satisfied under the system dynamics (4.1):

$$h(s_{t+1}) \geq p^T \Big(f(s_t) + g(s_t)a_t + \mu_d(s_t) \Big) - k_\delta |p|^T \sigma_d(s_t) + q. \tag{4.7}$$

Therefore, the constraint in problem (4.6) ensures that:

$$\begin{aligned} &h(s_{t+1}) \geq (1-\eta)h(s_t) - \epsilon, \\ &p^T s_{t+1} + q \geq (1-\eta)(p^T s_t + q) - \epsilon, \\ &p^T s_{t+1} + q + \frac{\epsilon}{\eta} \geq (1-\eta)(p^T s_t + q + \frac{\epsilon}{\eta}). \end{aligned} \tag{4.8}$$

Define $h_\epsilon(s) = q + \frac{\epsilon}{\eta} + p^T s$, so that (4.8) simplifies to

$$h_\epsilon(s_{t+1}) \geq (1-\eta)h_\epsilon(s_t). \tag{4.9}$$

By Definition 2, the set C_ϵ defined by $h_\epsilon(s) = h(s) + \frac{\epsilon}{\eta} \geq 0$ is forward invariant under system dynamics (4.1). □

Intuitively, Lemma 3 states that when possible, u^{CBF} provides the *minimal* control intervention that maintains safety. However, if such a control does not exist (e.g. due to torque constraints), then the CBF policy provides the control that keeps the state as close as possible to the safe set. Furthermore, even with dynamics uncertainty, one can make high-probability statements about system safety using GP models with CBFs.

To summarize, the model-free RL policy $\pi^{RL}_{\theta_k}(a|s)$ proposes a control action that attempts to optimize long-term reward, but may be unsafe. Before deploying the RL policy, a CBF controller $u^{CBF}_k(s)$ filters the proposed control action and provides the *minimum* control intervention needed to ensure that the overall policy, $\pi_k(a|s)$, keeps the system state within the safe set. Essentially, the CBF policy $\pi^{CBF}_k(a|s, \pi^{RL}_{\theta_k})$ "projects" the RL policy $\pi^{RL}_{\theta_k}(a|s)$ into the set of safe policies. In the case of an autonomous car, this action may enforce a safe distance between nearby cars, regardless of the action proposed by the RL policy.

Remark 7. Note that (4.6) includes a CBC constraint for a single CBF. However, one can easily combine multiple CBF constraints into the optimization to define polytopic safe regions.

The concept behind controller (4.4) is akin to shielded RL [10, 65], since the CBF controller compensates for the RL policy to ensure safety. However, this naive application of shielding may run into issues when integrated into many RL frameworks, because it does not account for the interplay between $u^{RL}_{\theta_k}$ and u^{CBF}. This may lead to distortion in the policy gradient. A more intuitive description of this issue is that the RL policy being updated, $\pi^{RL}_{\theta_k}(a|s)$, is *not* the policy deployed on the agent, $\pi_k(a|s)$. For example, suppose that in an autonomous driving task, the RL policy inadvertently proposes to collide with an obstacle. The CBF policy compensates to drive the car around the obstacle. The next learning iteration should update the policy around the safe deployed policy $\pi_k(a|s)$, rather than the unsafe policy $\pi^{RL}_{\theta_k}(a|s)$ (which would have led to an obstacle collision). However, RL algorithms are designed to update around their original policy, $\pi^{RL}_{\theta_k}(a|s)$ (typically limited to a trust region), as illustrated in Figure 4.2a.

4.2 Guiding Exploration in RL through CBFs

In order to achieve safe *and efficient* learning, one should learn from the deployed policy π_k, since it operates in the safe region C, rather than learning around $\pi^{RL}_{\theta_k}$, which may operate in an unsafe, irrelevant area of state space. The *RL-CBF* algorithm described in this section incorporates this goal. However, before jumping into the algorithm, let us go on a slight tangent to describe the issues arising from filtering the reinforcement learning policy, as done with policy (4.4).

4.2.1 Policy Gradient Distortion

Recall that the reinforcement learning algorithm is designed to optimize the policy by updating the parameters θ,

$$\theta_{k+1}^{RL} = \theta_k^{RL} + \alpha \nabla_\theta J(\theta_k^{RL}) \quad \text{such that} \quad J(\pi_{\theta_{k+1}}^{RL}) > J(\pi_{\theta_k}^{RL}), \tag{4.10}$$

where the policy gradient $\nabla_\theta J(\theta)$ is defined as,

$$\begin{aligned} \nabla_\theta J(\theta^{RL}) &= \mathbb{E}_{\tau \sim \pi_\theta^{RL}} \Big[\nabla_\theta \log \pi_\theta^{RL}(\tau) Q^{\pi_\theta^{RL}}(\tau) \Big] \\ &= \sum_{s \in S} d^{\pi_\theta^{RL}}(s) \sum_{a \in A} \pi_\theta^{RL}(a|s) Q^{\pi_\theta^{RL}}(s,a) \frac{\nabla_\theta \pi_\theta^{RL}(a|s)}{\pi_\theta^{RL}(a|s)}. \end{aligned} \tag{4.11}$$

However, note that the policy gradient is computed assuming that $\tau \sim \pi_\theta^{RL}$. This can pose an issue since the deployed policy $\pi = \pi_\theta^{RL} * \pi^{CBF}$. Therefore, the actual computed policy gradient will look like the following,

$$\begin{aligned} \nabla_\theta J(\theta^{RL}) &= \mathbb{E}_{\tau \sim \pi} \Big[\nabla_\theta \log \pi_\theta^{RL}(\tau) Q^{\pi_\theta^{RL}}(\tau) \Big] \\ &= \sum_{s \in S} d^{\pi}(s) \sum_{a \in A} \pi(a|s) Q^{\pi_\theta^{RL}}(s,a) \frac{\nabla_\theta \pi_\theta^{RL}(a|s)}{\pi_\theta^{RL}(a|s)} \\ &= \sum_{s \in S} d^{\pi_\theta^{RL} * \pi^{CBF}}(s) \sum_{a \in A} \big(\pi_\theta^{RL}(a|s) * \pi^{CBF}(a|s) \big) Q^{\pi_\theta^{RL}}(s,a) \frac{\nabla_\theta \pi_\theta^{RL}(a|s)}{\pi_\theta^{RL}(a|s)}, \end{aligned} \tag{4.12}$$

which implies that $J(\pi_{\theta_{k+1}}^{RL} * \pi_k^{CBF}) > J(\pi_{\theta_k}^{RL} * \pi_k^{CBF})$. The distinction is subtle, but clearly expressions (4.11) and (4.12) are not equivalent due to the convolution $\pi_\theta^{RL} * \pi^{CBF}$. Both represent the policy gradient for the RL policy π_θ^{RL}, but expression (4.12) considers the fact that trajectories are drawn from the safe policy π. To show the implications of this difference, suppose the policy gradient (4.12) is used to update the RL policy at every iteration while using the policy (4.4). At every iteration, the following would hold,

$$J(\pi_{\theta_{k+1}}^{RL}) - J(\pi_k) = J(\pi_{\theta_{k+1}}^{RL}) - J(\pi_{\theta_k}^{RL} * \pi_k^{CBF}), \tag{4.13}$$

which is not ideal due to the inclusion of π_k^{CBF}.

4.2.2 Leveraging the Policy Gradient Distortion

To account for the distortion in the policy gradient, inspired by (4.12), let us define the following *guided* policy,

$$\begin{aligned}\pi_k^{filter}(a|s) &= \pi_0^{CBF}(a|s) * \ldots * \pi_k^{CBF}(a|s)\\ \pi_{\theta_k}^{prop}(a|s) &= \pi_{\theta_k}^{RL}(a|s) * \pi_{k-1}^{filter}(a|s)\\ \pi_k(a|s) &= \pi_{\theta_k}^{RL}(a|s) * \pi_k^{filter}(a|s),\end{aligned} \tag{4.14}$$

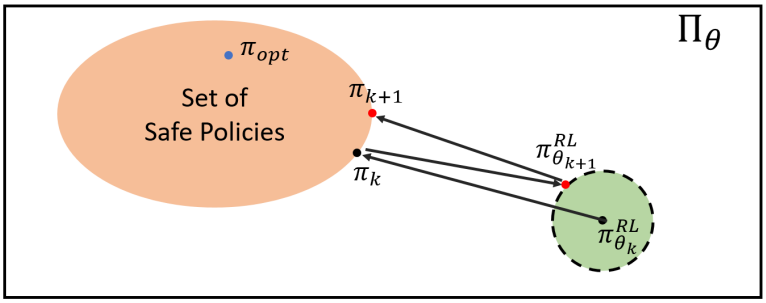
where π_k is the safe deployed policy. See Equation (4.24) for the deterministic analogue to (4.14), defining controllers drawn from these distributions. Recall that $\pi_k^{CBF}(a|s) = \mathbb{1}(a = u_k^{CBF}(s))$ with $\mathbb{1}(x)$ being the indicator function. Although $u_k^{CBF}(s)$ is a deterministic controller, for convenience, I will refer to it as policy $\pi_k^{CBF}(a|s)$ for the rest of this chapter.

As a concrete example, consider an initial RL-based policy $\pi_{\theta_0}^{RL}(a|s)$ (for iteration $k = 0$). The CBF policy $\pi_0^{CBF}(a|s)$ is determined from (4.6) to obtain $\pi_0(a|s) = \pi_{\theta_0}^{RL}(a|s) * \pi_0^{CBF}(a|s)$. Then for every following policy iteration, I define the overall policy to incorporate all previous CBF policies, as in (4.14). Using this policy (as opposed to the policy in (4.4)), at every iteration the following difference can be defined

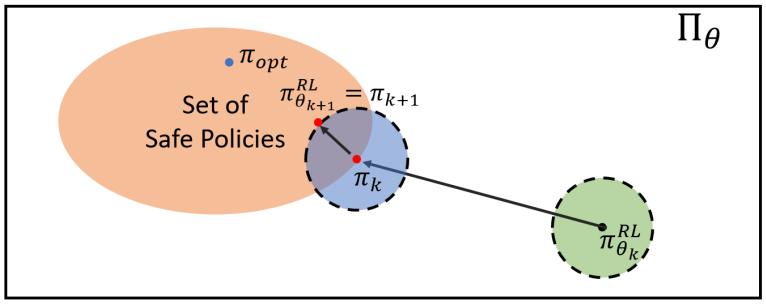
$$J(\pi_{\theta_{k+1}}^{prop}) - J(\pi_k) = J(\pi_{\theta_{k+1}}^{RL} * \pi_k^{filter}) - J(\pi_{\theta_k}^{RL} * \pi_k^{filter}), \tag{4.15}$$

whose performance is optimized by the computed policy gradient (4.12) (see proof of Theorem 3 for more details on how the modified policy π_k in (4.14) leverages these computed policy gradients).

The dependence of the overall policy π_k (in (4.14)) on all prior CBF policies (see Figure 4.1b) is critical to enhancing learning efficiency and avoiding distortion of the policy gradients. Defining the policy in this fashion leads to policy updates around the previously *deployed* policy, which adds to the efficiency of the learning process by encouraging the policy to operate in safe areas of the state space. This idea is illustrated in Figure 4.2b. The intuition is that at iteration $k = 0$, the RL policy *proposed* actions $u_{\theta_0}^{RL}(s)$, but it *took safe* actions $u_{\theta_0}^{RL}(s) + u_0^{CBF}(s)$. To update the policy based on the safe actions, the "guided" RL policy $\pi_{\theta_k}^{prop}$ at the next iteration ($k = 1$) should be $\pi_{\theta_1}^{RL}(a|s) * \pi_0^{CBF}(a|s)$ (i.e. $\pi_0^{CBF}(a|s)$ is now part of the RL policy). To ensure safety, this guided RL policy is then filtered by the CBF policy $\pi_1^{CBF}(a|s)$. For example at policy iteration k, one can consider $\pi_{\theta_k}^{RL} * \pi_0^{CBF} * \ldots * \pi_{k-1}^{CBF}$ to be the guided RL policy (proposing potentially unsafe actions), which is rendered safe by π_k^{CBF}.



(a)



(b)

Figure 4.2: Illustration of policy iteration process, where the goal is to learn the optimal safe policy, π_{opt}. *(a)* Policy optimization with barrier-compensating policy. Next policy is updated around the previous RL policy $\pi^{RL}_{\theta_k}$; *(b)* Policy optimization with barrier-guided policy. Next policy is updated around previous deployed policy π_k.

To ensure safety after incorporating all prior CBF policies, they must be included into the governing QP:

$$
\begin{aligned}
(a_t, \epsilon) = \underset{a\in A, \epsilon\in\mathbb{R}_+}{\operatorname{argmin}} \ & \|a\|_2 + K_\epsilon \epsilon \\
\text{s.t.} \quad & p^T f(s_t) + p^T g(s_t)\Big(u^{RL}_{\theta_k}(s_t) + \sum_{j=0}^{k-1} u^{CBF}_j(s_t) + a\Big) \\
& + p^T \mu_d(s_t) - k_\delta |p|^T \sigma_d(s_t) + q \geq (1-\eta) h(s_t) - \epsilon \\
& a^i_{low} \leq a^i + u^{RL,i}(s_t) + \sum_{j=0}^{k-1} u^{CBF,i}_j(s_t) \leq a^i_{high} \\
& \qquad\qquad\qquad\qquad \text{for } i = 1, ..., m.
\end{aligned}
\tag{4.16}
$$

The solution to (4.16) defines the CBF policy $\pi^{CBF}_k(a|s) = \delta(a - u^{CBF}_k(s))$ in (4.14), which ensures safety by satisfying the control barrier condition (4.3).

Let $\epsilon^{max} = \max_{s\in C} \epsilon$ from (4.16) represent the largest violation of the barrier condition for any $s \in C$.

Theorem 3. *Using the policy* $\pi_k(s)$ *from* (4.14)*, if there exists a solution to problem* (4.16) *such that* $\epsilon^{max} = 0$*, then the safe set* C *is forward invariant with probability* $(1-\delta)$*. If* $\epsilon^{max} > 0$*, but the solution to problem* (4.16) *satisfies* $\epsilon \leq \epsilon^{max}$ *for all* $s \in C_\epsilon$*, then the policy will render the larger set* C_ϵ *forward invariant with probability* $(1-\delta)$*.*

Furthermore, if one uses TRPO for the RL algorithm, then the guided RL policy $\pi^{prop}_{\theta_k}(a|s)$ *from* (4.14) *achieves the performance guarantee* $J(\pi^{prop}_{\theta_k}) \geq J(\pi_{k-1}) - \frac{2\lambda\gamma}{(1-\gamma)^2}\delta_\pi$*, where* $\lambda = \max_s |\mathbb{E}_{a\sim\pi^{prop}_{\theta_k}}[A_{\pi_{k-1}}(s,a)]|$*.*

Proof. The first part of the theorem follows directly from Definition 2 and Lemma 3. The only difference from Lemma 3 is that the control includes the RL policy and all *previous* CBF controllers $(u_0^{CBF}, ..., u_{k-1}^{CBF})$.

To prove the performance bound in the second part of the theorem, consider the property of the advantage function from equation (4.17) below:

$$J(\pi_k) = J(\pi_{k-1}) + \mathbb{E}_{\tau\sim\pi_k}\Big[\sum_{t=0}^{\infty}\gamma^t A_{\pi_{k-1}}(s_t,a_t)\Big], \tag{4.17}$$

where $s_{t+1} \sim P(s_{t+1}|s_t,a_t)$. As derived in [148], one can then obtain the following inequality:

$$\begin{aligned} J(\pi_k) \geq J(\pi_{k-1}) + \frac{1}{1-\gamma}\mathbb{E}_{\substack{s_t\sim\pi_{k-1}\\ a_t\sim\pi_k}}\Big[\sum_{t=0}^{\infty}\gamma^t A_{\pi_{k-1}}(s_t,a_t) \\ - \frac{2\gamma\lambda}{1-\gamma}D_{TV}(\pi_{k-1},\pi_k)\Big], \end{aligned} \tag{4.18}$$

where $D_{TV}(\pi_{k-1},\pi_k)$ is the total variational distance between policies π_{k-1} and π_k, and $\lambda = \max_s |\mathbb{E}_{a\sim\pi_k}[A_{\pi_{k-1}}(s,a)]|$. Note that the CBF controllers are all deterministic, so let us redefine $u_{k-1}^{barrier} = \sum_{j=0}^{k-2} u_j^{CBF} + u_{k-1}^{CBF} = \sum_{j=0}^{k-1} u_j^{CBF}$. Based on this definition and equation (15), rewrite/define the following controllers:

$$\begin{aligned} u_{k-1}(s) &= u^{RL}_{\theta_{k-1}}(s) + u^{barrier}_{k-1}(s), \\ \pi_{k-1}(a|s) &= \pi^{RL}_{\theta_{k-1}}(a - u^{barrier}_{k-1}(s) \mid s), \end{aligned} \tag{4.19}$$

$$
\begin{aligned}
u_k^{prop}(s) &= u_{\theta_k}^{RL}(s) + u_{k-1}^{barrier}(s), \\
\pi_k^{prop}(a|s) &= \pi_{\theta_k}^{RL}(a - u_{k-1}^{barrier}(s) \mid s).
\end{aligned} \tag{4.20}
$$

Plug in the above relations for π_{k-1} and π_k^{prop} into inequality (4.18), to obtain the following bound (plug in π_k^{prop} for π_k):

$$
\begin{aligned}
J(\pi_k^{prop}) \geq J(\pi_{k-1}) + \frac{1}{1-\gamma} \mathbb{E}_{\substack{s_t \sim \pi_{k-1} \\ a_t \sim \pi_k^{prop}}} \Big[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{k-1}}(s_t, a_t) \\
- \frac{2\gamma\lambda}{1-\gamma} D_{TV}(\pi_{\theta_{k-1}}^{RL}(a - u_{k-1}^{barrier}), \pi_{\theta_k}^{RL}(a - u_{k-1}^{barrier})) \Big],
\end{aligned} \tag{4.21}
$$

where the policies' dependence on the state s is dropped for compactness. Due to the shift invariance of the total variational distance, D_{TV}, this simplifies to:

$$
\begin{aligned}
J(\pi_k^{prop}) \geq J(\pi_{k-1}) + \frac{1}{1-\gamma} \mathbb{E}_{\substack{s_t \sim \pi_{k-1} \\ a_t \sim \pi_k^{prop}}} \Big[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{k-1}}(s_t, a_t) \\
- \frac{2\gamma\lambda}{1-\gamma} D_{TV}(\pi_{\theta_{k-1}}^{RL}, \pi_{\theta_k}^{RL}) \Big].
\end{aligned} \tag{4.22}
$$

Because π_{k-1} is a feasible point of the TRPO optimization problem (4) with objective value 0, the solution π_k^{prop} satisfies the following:

$$
\mathbb{E}_{\substack{s_t \sim \pi_{k-1} \\ a_t \sim \pi_k^{prop}}} \Big[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{k-1}}(s_t, a_t) \Big] \geq 0.
$$

Since the optimization problem (4) specifies the bound $D_{TV}(\pi_{\theta_{k-1}}^{RL}, \pi_{\theta_k}^{RL}) \leq \delta_\pi$, then it follows that:

$$
J(\pi_k^{prop}) \geq J(\pi_{k-1}) - \frac{2\lambda\gamma}{(1-\gamma)^2} \delta_\pi, \tag{4.23}
$$

where $\lambda = \max_s |\mathbb{E}_{a \sim \pi_k^{prop}}[A_{\pi_{k-1}}(s, a)]|$. The realization of the policy $\pi_k^{prop}(a|s)$ is:

$$
u_k^{prop}(s) = u_{\theta_k}^{RL}(s) + u_{k-1}^{barrier}(s) = u_k(s) - u_k^{CBF}(s).
$$

Therefore, utilizing the policy $u_k(s) - u_k^{CBF}(s)$, one can obtain the performance bound in equation (4.23).

□

RL-CBF provides high-probability safety guarantees during the learning process *and* guides the RL policy exploration (as exemplified by the performance guarantees with TRPO). If there is no uncertainty in the dynamics, then safety is guaranteed with probability 1. Note that the performance guarantee in Theorem 3 is for policy $\pi_{\theta_k}^{prop}(a|s)$, which is not the deployed policy, $\pi_k(a|s)$. However, this does not pose a significant issue, since $u_k^{CBF}(s)$ rapidly decays to 0 with increasing iterations. This limit is reached because the guided RL policy quickly learns to operate in the safe region, so the CBF controller $u_k^{CBF}(s)$ becomes inactive.

4.2.3 Computationally Efficient Algorithm

This section describes an efficient algorithm to implement the framework described above, since a naive approach would be too computationally expensive in many cases. To see this, let us write out the realized control law drawn from the stochastic policy π_k in (4.14):

$$\begin{aligned} u_k(s) = u_{\theta_k}^{RL}(s) + \sum_{j=0}^{k-1} u_j^{CBF}(s, u_{\theta_0}^{RL}, ..., u_{\theta_{j-1}}^{RL}) \\ + u_k^{CBF}(s, u_{\theta_k}^{RL} + \sum_{j=0}^{k-1} u_j^{CBF}). \end{aligned} \tag{4.24}$$

The first term in (4.24) may be represented by a neural network that is parameterized by θ_k, which has a standard implementation. The third term is just a quadratic program with dependencies on the other terms; it does not pose a computational burden. *However*, the summation in the 2nd term poses a challenge, since *every* term in $\sum_{j=0}^{k-1} u_j^{CBF}(s, u_{\theta_0}^{RL}, ..., u_{\theta_{j-1}}^{RL})$ depends on a different previous RL controller $u_{\theta_j}^{RL}$. Therefore, one would need to store $k-1$ neural networks corresponding to each previous RL controller. In addition, this would require solving $k-1$ separate QPs in sequence to evaluate each CBF controller. Such a brute-force implementation would be impractical.

To overcome this issue, approximate $u_{\phi_k}^{bar}(s) \approx \sum_{j=0}^{k-1} u_j^{CBF}(s, u_{\theta_0}^{RL}, ..., u_{\theta_{j-1}}^{RL})$, where $u_{\phi_k}^{bar}$ is a feedforward neural network parameterized by ϕ. Thus, at each policy iteration, the neural network $u_{\phi_k}^{bar}(s)$ is fit to data of $\sum_{j=0}^{k-1} u_j^{CBF}(s, u_{\theta_0}^{RL}, ..., u_{\theta_{j-1}}^{RL})$

collected from trajectories of the previous policy iteration. Then one obtains the controller:

$$u_k(s) = u^{RL}_{\theta_k}(s) + u^{bar}_{\phi_k}(s) + u^{CBF}_k(s, u^{RL}_{\theta_k} + u^{bar}_{\phi_k}). \tag{4.25}$$

Note that even with this approximation, *safety with probability* $(1-\delta)$ *is still guaranteed*. This is because the above approximation only affects the guided RL policy/controller $u^{RL}_{\theta_k}(s) + \sum_{j=0}^{k-1} u^{CBF}_j(s, u^{RL}_{\theta_0}, ..., u^{RL}_{\theta_{j-1}})$. The CBF controller $u^{CBF}_k(s, u^{RL}_{\theta_k} + u^{bar}_{\phi})$ still solves (4.16), which provides the safety guarantees in Theorem 3 by satisfying the CBF condition (4.3). Furthermore, it is now necessary to store only two NNs and solve one QP for the controller. The tradeoff is that the performance guarantee in Theorem 3 does not necessarily hold with this approximation. The algorithm is outlined in Algorithm 2.

4.3 Simulation Results: RL-CBF

I implement two versions of the RL-CBF algorithm with existing model-free RL algorithms: TRPO-CBF, derived from TRPO [148], and DDPG-CBF, derived from DDPG [109]. DDPG is an off-policy actor-critic method that computes the policy gradient based on sampled trajectories and an estimate of the action-value function. It alternately updates the action-value function and the policy as it samples more and more trajectories. TRPO is an on-policy policy gradient method that maximizes a surrogate loss function, which serves as an approximate lower bound on the true loss function. It adapts gradient step sizes to ensure that the next parameter update falls within a "trust region." The code for these examples can be found at [2].

4.3.1 Inverted Pendulum

The RL-CBF algorithm is first applied to the control of a simulated inverted pendulum from the OpenAI gym environment (*pendulum-v0*), which has mass m and length, l, and is actuated by torque, u. The safe region is defined to be $\theta \in [-1, 1]$ radians, and define the reward function $r = \theta^2 + 0.1\dot{\theta}^2 + 0.001u^2$ to learn a controller that keeps the pendulum upright. The true system dynamics are defined as follows,

$$\begin{aligned} \theta_{t+1} &= \theta_t + \dot{\theta}_t \delta t + \frac{3g}{2l}\sin(\theta_t)\delta t^2 + \frac{3}{ml^2}u\delta t^2, \\ \dot{\theta}_{t+1} &= \dot{\theta}_t + \frac{3g}{2l}\sin(\theta_t)\delta t + \frac{3}{ml^2}u\delta t, \end{aligned} \tag{4.26}$$

with torque limits $u \in [-15, 15]$, and $m = 1,\ l = 1$. To introduce model uncertainty, the nominal model assumes $m = 1.4,\ l = 1.4$ (40% error in model parameters).

Algorithm 2 RL-CBF algorithm

Initialize RL Policy π_0^{RL}, state $s_0 \sim \rho_0$, measurement array $\hat{D}$, action array $\hat{A}$
for $t = 1, \ldots, T$ **do**
  Sample (but do not deploy) control $u_{\theta_0}^{RL}(s_t)$
  Solve for $u_0^{CBF}(s_t)$ from optimization problem (4.16)
  Deploy controller $u_0(s_t) = u_{\theta_0}^{RL}(s_t) + u_0^{CBF}(s_t)$
  Store state-action pair (s_t, u_0^{CBF}) in $\hat{A}$
  Observe (s_t, u_0, s_{t+1}, r_t) and store in $\hat{D}$
end for
Collect Episode Reward, $\sum_{t=1}^{T} r_t$
Update GP model using (2.14) and measurements $\hat{D}$
Set $k = 1$ (representing k^{th} policy iteration)
while $k <$ Episodes **do**
  Do policy iteration using RL algorithm based on previously observed episode/rewards to obtain $\pi_{\theta_k}^{RL}$
  Train $u_{\phi_k}^{bar}$ to approximate prior CBF controllers $(u_{\phi_k}^{bar} = u_0^{CBF} + ... + u_{k-1}^{CBF})$ using $\hat{A}$
  Initialize state $s_0 \sim \rho_0$
  for $t = 1, \ldots, T$ **do**
    Sample control $u_{\theta_k}^{RL}(s_t) + u_{\phi_k}^{bar}(s_t)$
    Solve for $u_k^{CBF}(s_t)$ from problem (4.16)
    Deploy controller $u_k(s_t) = u_{\theta_k}^{RL}(s_t) + u_{\phi_k}^{bar}(s_t) + u_k^{CBF}(s_t)$.
    Store state-action pair $(s_t, u_{\phi_k}^{bar} + u_k^{CBF})$ in $\hat{A}$
    Observe (s_t, u_k, s_{t+1}, r_t) and store in $\hat{D}$
  end for
  Collect Episode Reward, $\sum_{t=1}^{T} r_t$
  Update GP model using (2.14) and measurements $\hat{D}$
  $k = k + 1$
end while
return $\pi_{\theta_k}^{RL}, u_{\phi_k}^{bar}, u_k^{CBF}$ ▷ Overall policy composed of all 3 subcomponents

Figure 4.3 compares the accumulated reward achieved during each episode using TRPO, DDPG, TRPO-CBF, and DDPG-CBF. The two RL-CBF algorithms converge near the optimal solution very rapidly, and significantly outperform the corresponding baseline algorithms without the CBFs. Note that TRPO and DDPG *sometimes* converge on a high-performance policy (comparable to TRPO-CBF and DDPG-CBF), though this occurs less reliably and more slowly, resulting in the poorer learning curves. More importantly, the RL-CBF policies maintain safety (i.e. never leave the safe region) throughout the learning process, as also seen in Figure 4.3. In contrast, TRPO and DDPG severely violate safety while learning the

optimal policy.

Figure 4.4 shows the pendulum angle during a representative trial under the first policy versus the last learned policy deployed for TRPO-CBF and DDPG-CBF. For the first policy iteration, the pendulum angle is maintained near the edge of the safe region – the RL algorithm has proposed a poor control action so the CBF controller provides the minimum intervention necessary to keep the system safe. By the last iteration though, the CBF controller is completely inactive ($u^{CBF} = 0$), since the guided RL policy ($u^{RL}_{\theta_k}(s) + u^{bar}_{\phi_k}(s)$) is already safe.

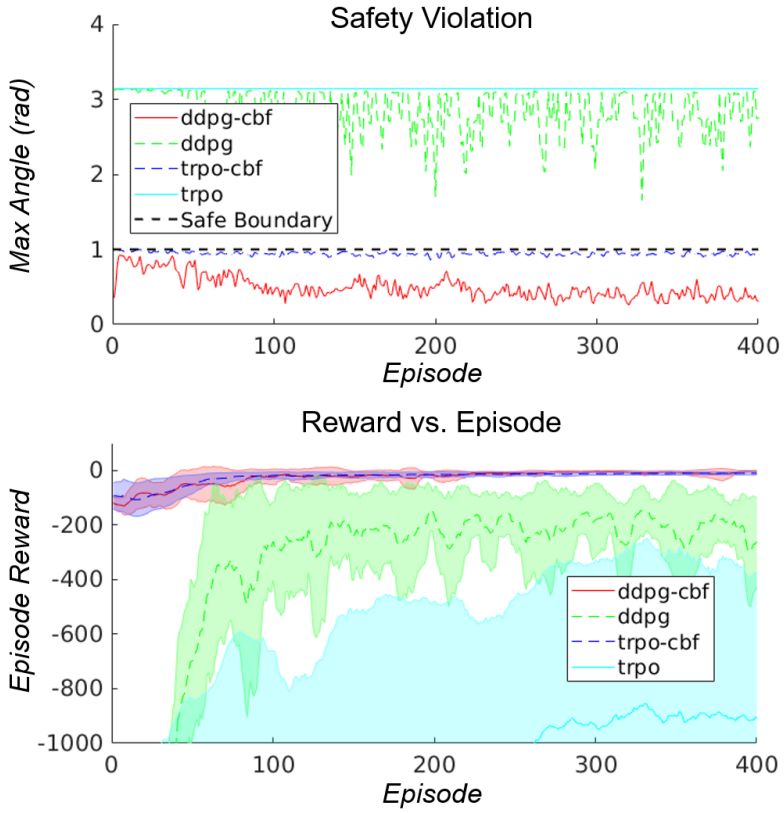


Figure 4.3: (Top) Maximum angle (rad) of the pendulum throughout each episode. Values above the dashed black line represent exits from the safe set at some point during the episode. (Bottom) Comparison of accumulated reward from inverted pendulum problem using TRPO, DDPG, TRPO-CBF, and DDPG-CBF.

4.3.2 Simulated Car Following

Consider a chain of five cars following each other on a straight road. The RL policy controls the acceleration/deceleration of the 4^{th} car in the chain, and the goal is to train the policy to maximize fuel efficiency during traffic congestion while avoiding collisions. Each car utilizes the dynamics shown in equation (4.27), and the controlled car attempts to optimize the reward function (4.28). The car dynamics

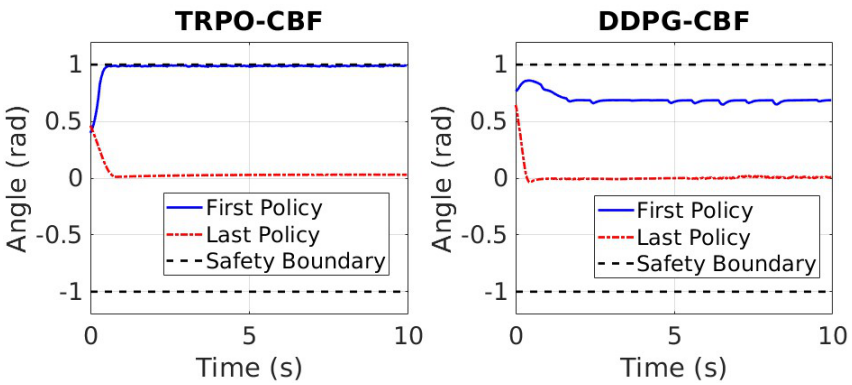


Figure 4.4: Representative pendulum trajectory (angle vs. time) using first policy vs. last policy. The left plot and right plot show results from TRPO-CBF and DDPG-CBF, respectively. The trajectory for the first policy (blue) goes to edge of the safe region and stays there, while the trajectory for the last policy (red) quickly converges to the upright position.

and reward function are inspired by previous work [87].

$$\begin{bmatrix} \dot{s}^{(i)} \\ \dot{v}^{(i)} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -k_d \end{bmatrix} \begin{bmatrix} s^{(i)} \\ v^{(i)} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a \qquad k_d = 0.1. \tag{4.27}$$

$$\begin{aligned} r &= -\sum_{t=1}^{T} \left[v_t^{(4)} \max((a_t^{(4)}), 0) + \sum_{i=3}^{4} G_i\Big(\frac{500}{s_t^{(i)} - s_t^{(i+1)}}\Big)\right]. \\ G_m(x) &= \begin{cases} |x| & \text{if } s^{(m)} - s^{(m+1)} \leq 3 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4.28}$$

The first term in the reward optimizes fuel efficiency, while the other term encourages the car to maintain a 3-meter distance from the other cars (soft constraint). For the RL-CBF controllers, the CBF enforces a 2-meter safe distance between cars (hard constraint).

The 4^{th} car has access to every other cars' position, velocity, and acceleration, but it only has a crude model of its own dynamics ($k_d = 0$) and an inaccurate model of the drivers behind and in front of it. In addition, Gaussian noise is added to the acceleration of each car. The idea is that the 4^{th} car can use its crude model to guarantee safety with high probability, *and* improve fuel efficiency by slowly building and leveraging an implicit model of the other drivers' behaviors.

From Figure 4.5, it is shown that there were no safety violations between cars during the simulated experiments when using either of the RL-CBF controllers. When using TRPO and DDPG alone without CBF safety, almost all trials had collisions,

even in the later stages of learning. Furthermore, as seen in Figure 4.5, TRPO-CBF learns faster and outperforms TRPO (DDPG-CBF also outperforms DDPG, though neither algorithm converged on a high-performance controller in the experiments). It is important to note that in *some* experiments, TRPO finds a comparable controller to TRPO-CBF, but this is often not the case due to randomness in seeds (relating to the variance issue discussed in Chapter 3).

Although DDPG and DDPG-CBF failed to converge on a good policy, Figure 4.5 shows that DDPG-CBF (and TRPO-CBF) always maintained a safe controller. This is a crucial benefit of the RL-CBF approach, as it guarantees safety independent of the system's learning performance and regardless of misspecified rewards.

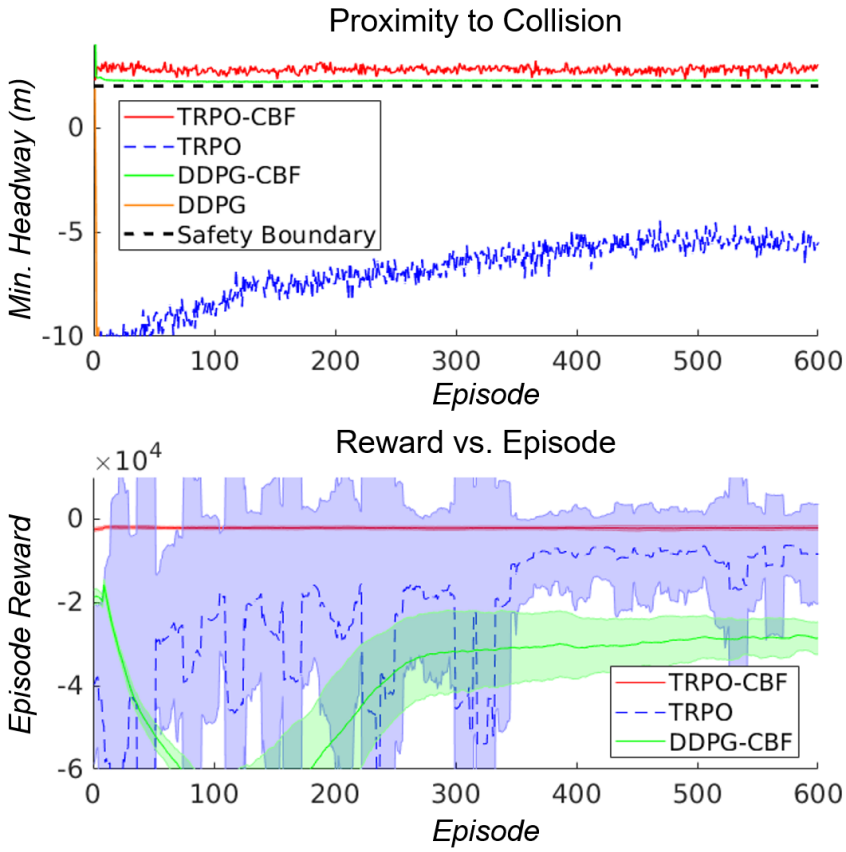


Figure 4.5: (Top) Minimum headway between cars during each learning episode using DDPG, TRPO, DDPG-CBF, and TRPO-CBF. Values below the dashed black line represent exits from the safe set, and values below 0 represent collisions. The curve for DDPG has high negative values throughout learning, and is not seen. (Bottom) Comparison of reward over multiple episodes from car-following problem using TRPO, TRPO-CBF, and DDPG-CBF (DDPG is excluded because it exhibits very poor performance).

4.4 Robust Safety in Uncertain, Multi-Agent Interactions

So far, this chapter has looked at guaranteeing safety and guiding exploration in RL using discrete-time control barrier functions. However, the results in the previous sections were limited to affine control barrier functions, as more complex CBFs might not allow a quadratic program formulation for the optimization problem (4.16). Furthermore, the Gaussian process model used to learn the dynamics uncertainties, d, considers each element independently (i.e. does not model interdependencies between the different components of d). These restrictions can be limiting in many scenarios. Therefore, this section looks at overcoming these two restrictions, specifically in the context of guaranteeing safety during uncertain, multi-agent navigation.

Collision-free robot navigation in natural multi-agent environments is vital for a myriad of robotic applications, such as self-driving cars, navigation in crowds, etc. However, placing robots in rapidly evolving, uncertain environments introduces many challenges in guaranteeing safety [25, 64, 91, 98, 169]. Uncertainty in the prediction of other agents' trajectories is inevitable (human trajectories remain notoriously difficult to predict and are highly stochastic [23]), and robots must learn and account for this uncertainty to ensure safe operation. Therefore, the overarching goal of this section is to (1) learn uncertainty bounds *offline and online* from agents' observed trajectories, and (2) incorporate those uncertainty bounds into a multi-agent CBF [27] while maintaining computational efficiency of the underlying controller (i.e. a quadratic program).

The proposed approach focuses first on learning high-confidence polytopic bounds on the, *possibly coupled*, uncertainties in both the robot dynamics and other agents' dynamics. To achieve this, I utilize matrix-variate Gaussian processes (MVG) and optimize their hyperparameters *offline* from interaction data; this allows for the prediction of ellipsoidal uncertainty in the dynamics *online*, which can be converted to an uncertainty polytope given a desired confidence level. Using these polytopic bounds, a robust CBF can be formulated as a min-max optimization problem over the robot controls and the potential uncertainties, respectively. This min-max problem is then transformed into a quadratic program that can be efficiently solved to find a safe control action that is robust with respect to the estimated uncertainty. Figure 4.6 provides an overview of the proposed approach.

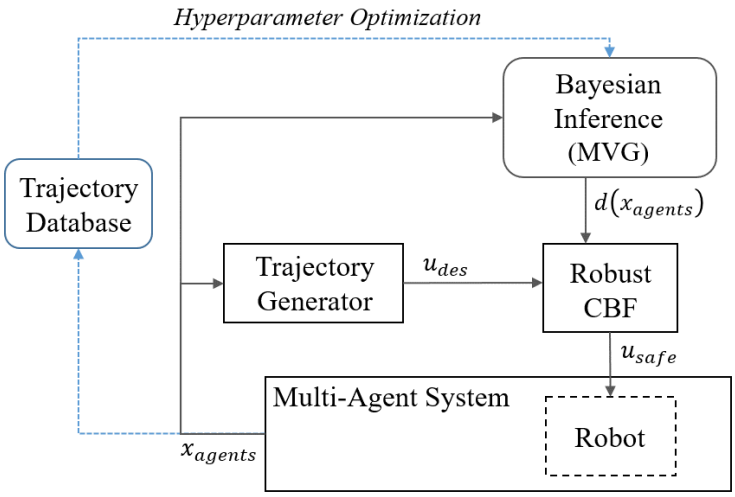


Figure 4.6: Diagram overviewing the control structure. The proposed approach guarantees safety by utilizing a Bayesian Inference Module to learn dynamic uncertainties, and handles them with the proposed Robust CBF module.

4.4.1 Multi-Agent Problem Setup

Let us consider the system dynamics (4.1) but now explicitly include position, p and velocity, v in the state,

$$s_{t+1} = \begin{bmatrix} p_{t+1} \\ v_{t+1} \\ z_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} f_p(s_t) \\ f_v(s_t) \\ f_z(s_t) \end{bmatrix}}_{f(s_t)} + \underbrace{\begin{bmatrix} g_p(s_t) \\ g_v(s_t) \\ g_z(s_t) \end{bmatrix}}_{g(s_t)} a_t + \underbrace{\begin{bmatrix} d_p(s_t) \\ d_v(s_t) \\ d_z(s_t) \end{bmatrix}}_{d(s_t)}, \tag{4.29}$$

where $p \in \mathbb{R}^2$, $v \in \mathbb{R}^2$, and $z \in \mathbb{R}^{n-4}$ denote position, velocity, and other states, respectively. Here, f_j, g_j, and d_j are real-valued functions, for $j \in \{p, v, z\}$. Assume that this system has relative degree 2 with respect to the positional output p; in discrete time, this directly implies that $g_p(x) = 0_{2\times 2}$. Similarly, let us represent the other agents within the multi-agent system with dynamics,

$$x^{(i)}_{t+1} = \begin{bmatrix} p^{(i)}_{t+1} \\ v^{(i)}_{t+1} \\ z^{(i)}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} f_p^{(i)}(s_t) \\ f_v^{(i)}(s_t) \\ f_z^{(i)}(s_t) \end{bmatrix}}_{f^{(i)}(s_t)} + \underbrace{\begin{bmatrix} d_p^{(i)}(s_t) \\ d_v^{(i)}(s_t) \\ d_z^{(i)}(s_t) \end{bmatrix}}_{d^{(i)}(s_t)}, \tag{4.30}$$

where $i \in \mathbb{N}$ indexes each of the other agents in the system. It is assumed that the control input for other agents are an (uncertain) function of their state at the given time, so control inputs are not included explicitly in (4.30).

Since the robot interacts with other unknown agents, it will be important to account for the uncertainties, $d, d^{(i)}$, when considering safety via CBFs. The rest of this

chapter assumes that each agent's current state s_t is perfectly observed, but that the uncertain dynamics $d(s_t)$ and $d^{(i)}(s_t)$ are unknown, but can be estimated over time.

4.4.2 Robust Multi-Agent CBF

Our goal is to ensure safety by defining/utilizing a discrete-time CBF for the system. Inspired by the multi-agent CBF proposed in [27] (discussed in Section 2.2), consider the following CBF for the discrete time system,

$$h(s) := \frac{\Delta p^T \Delta v}{\|\Delta p\|} + \sqrt{u_{max}(\|\Delta p\| - D_s)}\,, \tag{4.31}$$

where D_s is the collision margin, $\Delta p = p - p^{(i)}$ is the positional difference between the agents, and $\Delta v = v - v^{(i)}$ is the velocity difference between the agents. u_{max} will be defined later in (4.32), but intuitively it represents the robot's max acceleration in any potential collision direction.

Extending Multi-Agent CBF to Discrete-Time, Nonlinear Systems: This subsection shows that under proper assumptions, $h(s)$ defined in (4.31) is a discrete-time CBF for a restricted set of *discrete-time nonlinear* systems (4.29), (4.30). The tradeoff is the additional conservativeness in u_{max} introduced by the following assumption. Intuitively, this assumption ensures that the robot can accelerate in any direction relative to the other agents, as proved in Lemma 4.

Assumption 2. *Assume that for all $x \in C$, $g_v(x)$ is invertible and $\frac{\|\beta_v(x)\|}{\sigma_{min}(g_v(x))a_{max}} < 1$, where $\beta_v(x) = f_v(x) + d_v(x) - f_v^{(i)}(x) - d_v^{(i)}(x) - \Delta v_t$, $\sigma_{min}(g_v(x))$ is the minimum singular value of $g_v(x)$, and a_{max} is the maximum actuation authority in the dynamics* (4.29).

Remark 8. This assumption ensures controllability and places restrictions on the agent's dynamics with relation to its actuator authority. If a_{max} is large, the restriction is minimal, and vice-versa. As a simple example, a car at rest would *not* satisfy this assumption, though a moving car *would* likely satisfy this assumption (with a higher velocity corresponding to larger u_{max}).

Lemma 4. *Under Assumption 2, which places controllability restrictions on the dynamics, the expression* (4.31)*, defining set C, represents a discrete-time CBF for*

system (4.29)*, with*

$$u_{max} = \min_{x} \left[\sigma_{min}(g_v(x)) a_{max} - \|\beta_v(x)\| \right] > 0. \tag{4.32}$$

Proof. To prove Lemma 4, it suffices to show that the following statements are true:

- Set C defined by expression (4.31) is control invariant for the dynamics (4.29), given that the robot has acceleration authority in any direction of at least a_{max} for all $x \in C$.
- Under Assumption 2, $a_{max} > 0$ for all $x \in C$.

The proof of the **first point** relies on the same proof structure found in [27], with the main difference being that the problem is formulated with discrete-time (rather than continuous-time) dynamics.

Let $\Delta\hat{v}(x_t)$ denote the component of velocity $v(x_t)$ *in the direction of collision*.

$$\Delta\hat{v}(x_t) = \frac{\Delta p^T \Delta v}{\|\Delta p\|}. \tag{4.33}$$

Collision can be avoided if the robot can match the other agent's velocity (i.e. $\Delta v = 0$) by the time the robot reaches that agent. If one assumes that the robot can accelerate by a_{max} in any direction, it is guaranteed that the robot can achieve $\Delta v = 0$ within time $T_c = \frac{-\Delta\hat{v}(x_t)}{a_{max}}$. In the discrete-time formulation, the following condition implies collision avoidance:

$$\begin{aligned} &\Delta\hat{v}(x_t) T_c + \|\Delta p\| \geq D_s, \\ &-\frac{\Delta\hat{v}^2(x_t)}{a_{max}} + \|\Delta p\| \geq D_s, \\ &\left(\frac{\Delta p^T \Delta v}{\|\Delta p\|}\right)^2 \leq a_{max}(\|\Delta p\| - D_s). \end{aligned} \tag{4.34}$$

Note that this constraint is only active when two agents are moving closer to each other ($\Delta\hat{v} < 0$), and no constraint is needed when two agents are moving away from each other ($\Delta\hat{v} \geq 0$). Therefore, collision is always avoided under the following condition,

$$-\frac{\Delta p^T \Delta v}{\|\Delta p\|} \leq \sqrt{a_{max}(\|\Delta p\| - D_s)}. \tag{4.35}$$

Therefore, the function $h = \frac{\Delta p^T \Delta v}{\|\Delta p\|} + \sqrt{a_{max}(\|\Delta p\| - D_s)}$ is a discrete-time control barrier function, *under the assumptions made above, that the robot can accelerate by at least a_{max} in any direction.*

To address the **second point**: one must show that for all $x \in C$ and any unit vector $\hat{e}$, it holds that $\sup_{u \in \mathcal{U}} \|\left(v_{t+1}(x,u) - v_t(x)\right)^T \hat{e}\| \geq a_{max} > 0$.

$$\begin{aligned}
\sup_{u \in \mathcal{U}} \|\left(v_{t+1}(x,u) - v_t\right)^T \hat{e}\| &= \sup_{u \in \mathcal{U}} \|\left(\beta_v(x) + g_v(x)u\right)^T \hat{e}\| \\
&= \sup_{u \in \mathcal{U}} \|\hat{e}^T \beta_v(x) + \hat{e}^T g_v(x) u\| \\
&\geq \sup_{u \in \mathcal{U}} \|\hat{e}^T g_v(x) u\| - \|\hat{e}^T \beta_v(x)\| \\
&\geq \sup_{u \in \mathcal{U}} \|\hat{e}^T g_v(x) u\| - \|\beta_v(x)\| \\
&\geq \sigma_{min}(g_v(x)) u_{max} - \|\beta_v(x)\|.
\end{aligned} \tag{4.36}$$

This directly implies the following,

$$a_{max} = \min_x \Big[\sigma_{min}(g_v(x)) u_{max} - \|\beta_v(x)\|\Big], \tag{4.37}$$

which is positive based on Assumption 2. □

Incorporating Robustness into the CBF: While uncertainty in robot/environmental dynamics can be directly incorporated into the Control Barrier Condition (CBC) for simple discrete-time systems/constraints (as shown in the previous sections of this chapter), this is not the case for the multi-agent CBF with discrete-time dynamics.

Consider the CBF (4.31) and the dynamics defined in (4.29) and (4.30). Based on these, the following CBC can be computed with respect to each other agent i:

$$\begin{aligned}
CBC^{(i)}(s_t, a_t) = \Big\langle &\frac{f_p(s_t) + g_p(s_t)a_t + d_p(s_t) - f_p^{(i)}(s_t) - d_p^{(i)}(s_t)}{\|f_p(s_t) + g_p(s_t)a_t + d_p(s_t) - f_p^{(i)}(s_t) - d_p^{(i)}(s_t)\|}, \\
&f_v(s_t) + g_v(s_t)a_t + d_v(s_t) - f_v^{(i)}(s_t) - d_v^{(i)}(s_t)\Big\rangle + \\
&\sqrt{u_{max}(\|f_p(s_t) + g_p(s_t)a_t + d_p(s_t) - f_p^{(i)}(s_t) - d_p^{(i)}(s_t)\| - D_s)} + \\
&(\eta - 1)\sqrt{u_{max}(\|\Delta p_t\| - D_s)} + (\eta - 1)\frac{\Delta p_t^T \Delta v_t}{\|\Delta p_t\|}.
\end{aligned} \tag{4.38}$$

If one can **(a)** determine bounds on the dynamic uncertainties, d, in (4.29) and (4.30), and **(b)** compute control actions that satisfy $CBC(x,u) \geq 0$ in an online fashion,

then robust safety can be guaranteed by utilizing the multi-agent CBF. Ideally, one could incorporate (4.38) into an efficiently solvable program as follows,

$$\begin{aligned} a_t = \underset{a \in A}{\text{argmin}} \quad & \|a - a_{des}\|_2 \\ \text{s.t.} \quad & \min_{d(s_t)} CBC^{(i)}(s_t, a, d_t) \geq 0 \quad \forall\, i = 1, ..., M \\ & \text{where } d(s_t) \in \mathcal{D} \\ & \|a\|_2 \leq a_{max}\,, \end{aligned} \tag{4.39}$$

where a_{des} can be any, potentially unsafe, proposed control action (e.g. u_θ^{RL}), $\mathcal{D}$ is some bound on the uncertainty (further discussed in the following subsection), and M is the number of other agents. Note that the CBC constraint (4.38) in (4.39) is clearly not linear nor convex. Furthermore, the minimization over actions a must be done considering all $d \in \mathcal{D}$. Therefore, the resulting program is non-convex and cannot be efficiently solved at high frequency for adequate safety assurances. However, recall that the dynamical system has relative degree 2, which allows us to derive the following bound,

$$CBC(s_t, a_t, d_t) \geq k_c(s_t) - H_1(s_t)d_t - a_t^T H_2(s_t)d_t - H_3(s_t)a_t, \tag{4.40}$$

where the definitions of (k_c, H_1, H_2, H_3) are provided below.

$$H_1(1\times P) = \Big[-\frac{f_v(x)-f_v^h(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)},$$

$$-\frac{f_p(x)-f_p^h(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)},$$

$$\frac{f_v(x)-f_v^h(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)},$$

$$\frac{f_p(x)-f_p^h(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)}\Big]$$

$$H_2(M\times P) = \Big[-\frac{g_v(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)}\ ,\ 0\ ,$$

$$\frac{g_v(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)}\ ,\quad 0\ \Big] \tag{4.41}$$

$$H_3(1\times M) = \Big[-\frac{(f_p(x)-f_p^h(x))^T g_v(x)}{\|f_p(x)-f_p^h(x)\|+\zeta_p(x)+\zeta_p^h(x)}\ \Big]$$

$$k_c \ = \ \min\Big(\frac{(f_p(x)-f_p^h(x))^T(f_v(x)-f_v^h(x))}{\|f_p(x)-f_p^h(x)\| \ \pm\ \big(\zeta_p(x)+\zeta_p^h(x)\big)}\Big)+$$

$$\sqrt{a_{max}(\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)-D_s)}+$$

$$(\eta-1)\sqrt{a_{max}(\|\Delta p_t\|-D_s)}\ +\ (\eta-1)\frac{\Delta p_t^T\Delta v_t}{\|\Delta p_t\|}\ -$$

$$\frac{\zeta_p(x)\zeta_v(x)+\zeta_p(x)\zeta_v^h(x)+\zeta_v^h(x)\zeta_p^h(x)+\zeta_v(x)\zeta_p^h(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)}.$$

The derivation of bound (4.40) can be found at the end of this chapter (Appendix C). For the remainder of the section, the index i is dropped for notational convenience.

The following lemma allows us to utilize CBC bound (4.40) to obtain safety guarantees *under polytopic uncertainties*.

Lemma 5. *Suppose the uncertainty in the system dynamics, d, is bounded in the*

polytope $\{d \in \mathbb{R}^n \mid Gd \leq g\}$. *Then the action,* u*, obtained from solving the following optimization problem* (4.42) *robustly satisfies the CBC condition* (4.38) *(i.e. renders the set* C *forward invariant).*

$$\begin{aligned} \min_{a \in \mathcal{A}, \xi \in \mathbb{R}^{4n}_+} \quad & \|a - a_{des}\|_2 \\ s.t. \quad & H_3(s_t)a + \xi g \leq k_c(s_t) \\ & H_1(s_t) + a^T H_2(s_t) = \xi G \\ & \xi \geq 0 \\ & \|a\|_2 \leq a_{max} \quad \text{(actuation limits)} \ . \end{aligned} \tag{4.42}$$

Proof. The robust optimization problem (4.42) can be equivalently represented by the following optimization problem (i.e. (4.42) is the dual to (4.43) with no duality gap [33] where ξ is the dual variable):

$$\begin{aligned} \min_{a \in \mathcal{A}} \quad & \|a - a_{des}\|_2 \\ \text{s.t.} \quad & \forall d \in \{d \in \mathbb{R}^n \mid Gd \leq g\} \\ & H_1(s_t)d + a^T H_2(s_t)d + H_3(s_t)a \leq k_c(s_t) \\ & \|a\|_2 \leq a_{max} \quad \text{(actuation limits)} \ , \end{aligned} \tag{4.43}$$

where a is the decision vector, d is the uncertainty variable, and $\{d \in \mathbb{R}^n \mid Gd \leq g\}$ is the uncertainty bound. If the inequality in (4.43) is satisfied such that $H_1 d + u^T H_2 d + H_3 u \leq k_c$, then it follows directly from (4.40) that the CBC condition is satisfied for all d in the polytopic uncertainty set. Therefore, the set C is rendered forward invariant. □

This lemma shows that if $d(s_t)$ is bounded within a polytope, the robust multi-agent CBF (4.39) can be transformed into a quadratic program (4.42), which yields a computationally efficient way to provide robust guarantees of safety under robot and environment uncertainties. Hence, the following section examines the problem of learning accurate polytopic bounds on the uncertainty d in an online fashion.

4.4.3 Learning Uncertainty Bounds

To learn accurate confidence supports for the uncertainties d and $d^{(i)}$ (for all agents i) in an online manner, Matrix-Variate Gaussian Processes are utilized, which provide multivariate Gaussian distributions over the uncertainties, d and $d^{(i)}$. Recall from

Section 2.3 that given a set of N measurements, $s_{[N]}$, one can infer the posterior distribution $d(s_*)$ at target state s_* as follows,

$$\begin{aligned} d(s_*) \sim & \mathcal{N}\big(\hat{M}\ ,\ \hat{\Sigma} \otimes \hat{\Omega}\big) \\ & \hat{M} = K(s_*, s_{[N]})^T K(s_{[N]}, s_{[N]})^{-1} d_{[N]} \\ & \hat{\Sigma} = \kappa(s_*, s_*) - K(s_{[N]}, s_*)^T K(s_{[N]}, s_{[N]})^{-1} K(s_{[N]}, s_*) \\ & \hat{\Omega} = \Omega. \end{aligned} \tag{4.44}$$

Using (4.29), it follows that $d(s_t) = s_{t+1} - f(s_t) - g(s_t)a_t$. A similar relation holds based on (4.30) for other agents. Thus, given a sequence of measurements (s_t, a_t, s_{t+1}) over a horizon T, the uncertain variables, $d_{t-T}, ..., d_{t-1}$, are computed over that horizon. Then a distribution over the query point, d_t (i.e. next time point), can be inferred as described in Equation (4.44).

Optimizing Kernel Hyperparameters Offline: However, direct application of the MVG (4.44) to the multi-agent setup will be problematic without accurately trained model hyperparameters. This is easy to see by noting that the covariance, $\Sigma(s_{[N]}) \otimes \Omega$, does not depend on the observed values, $d_{[N]}$. Furthermore, the coupling between uncertainties, captured by Ω, is completely independent of the online measurements. Instead, much of the uncertainty prediction is baked into the kernel parameters, $\kappa(l, \sigma)$, and matrix Ω. Thus, to obtain accurate estimates of d, the MVG model parameters must be learned offline from data. In other words, some agents might behave predictably and others might behave more erratically, and hyperparameter optimization is necessary to capture these uncertainty profiles in the Bayesian inference process.

Based on the probability density function (2.16), the negative log-likelihood of a given set of training data X is

$$\begin{aligned} L(\mathbf{X}, \mathbf{Y};\ K, \Omega) = & -\ln p(\mathbf{X},\ \mathbf{Y}; K, \Omega) = \\ & \frac{Nn}{2}\ln(2\pi) + \frac{n}{2}\ln|K| + \frac{N}{2}\ln|\Omega| + \frac{1}{2}\,\mathbf{tr}[(K)^{-1} Y \Omega^{-1} Y^T], \end{aligned} \tag{4.45}$$

which is optimized (over Σ, Ω) using Stochastic Gradient Descent [40] (see hyperparameter optimization in Figure 4.6). Recall that N denotes the number of training samples in the batch, and n denotes the dimension of the output d. The optimization is initialized at different states to decrease the chance of getting stuck in poor local optima. The gradient expressions are shown in Equation (4.46) below. Projected gradient updates are used to find Ω, in order to enforce the condition that Ω must be positive definite.

$$\frac{dL}{dl} = \frac{n}{2}\mathbf{tr}\left(K^{-1}\frac{dK}{dl}\right) + \frac{1}{2}\mathbf{tr}\left(-K^{-1}\frac{dK}{dl}K^{-1}Y\Omega^{-1}Y^T\right)$$

$$\frac{dL}{d\sigma} = \frac{n}{2}\mathbf{tr}\left(K^{-1}\frac{dK}{d\sigma}\right) + \frac{1}{2}\mathbf{tr}\left(-K^{-1}\frac{dK}{dl}K^{-1}Y\Omega^{-1}Y^T\right) \tag{4.46}$$

$$\frac{dL}{d\Omega} = \frac{N}{2}\Omega^{-1} - \frac{1}{2}\Omega^{-1}Y^TK^{-1}Y\Omega^{-1}.$$

Converting GP Uncertainty to a Polytopic Bound: After learning the kernel parameters, one can compute the mean, $\mu_d = \hat{M}$, and variance, $\Sigma_d = \hat{\Sigma} \otimes \hat{\Omega}$, from data observed online based on the multivariate Gaussian Process (2.19). Then, the uncertainties should follow the distribution,

$$(d - \mu_d)^T\Sigma_d^{-1}(d - \mu_d) \sim \chi_n^2, \tag{4.47}$$

where χ_n^2 represents the chi-squared distribution with n degrees of freedom (equal to dimension of d). This expression allows us to obtain the confidence support,

$$(d - \mu_d)^T\Sigma_d^{-1}(d - \mu_d) \leq k_\delta \text{ with probability } 1 - \delta. \tag{4.48}$$

However, this set defines an *ellipsoid* over d rather than a polytope, which is required for the robust optimization. While the ellipsoidal constraint could be used directly, this approach would not lead to an efficiently solvable QP. A polytope is found by computing the minimum bounding box surrounding the uncertainty ellipsoid.

Lemma 6. *Suppose the robot/environment uncertainty can be described by an MVG model (described by the distribution* (4.47)*). With probability* $1 - \delta$*, the following polytopic bound on the uncertainty d holds:*

$$-\sqrt{k_\delta\lambda_i} + \upsilon_i^T\mu_d \leq \upsilon_i^T d \leq \sqrt{k_\delta\lambda_i} + \upsilon_i^T\mu_d, \tag{4.49}$$

where υ_i *and* λ_i *represent the eigenvectors and eigenvalues of* Σ_d*, respectively.*

Proof. Since Σ_d is a positive symmetric covariance matrix, the eigendecomposition of Σ_d exists: $\Sigma_d = \Psi^T\Lambda\Psi$, where Λ is a diagonal matrix containing positive

eigenvalues of Σ_d and Ψ is the orthogonal eigenvector matrix. Thus, (4.48) can be rewritten as follows:

$$\left[\Psi^T(d-\mu_d)\right]^T \Lambda^{-1}\left[\Psi^T(d-\mu_d)\right] \leq k_\delta \text{ w.p. } 1-\delta. \tag{4.50}$$

The left-hand side can be bounded as:

$$\begin{aligned}
&\left[\Psi^T(d-\mu_d)\right]^T \Lambda^{-1}\left[\Psi^T(d-\mu_d)\right] \\
&= \sum_i \left[\upsilon_i^T(d-\mu_d)\right]^T \lambda_i^{-1}\left[\upsilon_i^T(d-\mu_d)\right] \\
&\geq \left[\upsilon_i^T(d-\mu_d)\right]^T \lambda_i^{-1}\left[\upsilon_i^T(d-\mu_d)\right] \ \forall \ i=1,...,N,
\end{aligned} \tag{4.51}$$

where υ_i represent the eigenvectors of Σ_d contained in Ψ, and λ_i are the eigenvalues of Σ_d contained in Λ. Therefore, with probability at least $1-\delta$, the following relations hold, resulting in the polytopic bound,

$$\begin{aligned}
&\left[\upsilon_i^T(d-\mu_d)\right]^T \lambda_i^{-1}\left[\upsilon_i^T(d-\mu_d)\right] \leq k_\delta \ \forall \ i=1,...,N \\
&-\sqrt{k_\delta\lambda_i} + \upsilon_i^T\mu_d \leq \upsilon_i^T d \leq \sqrt{k_\delta\lambda_i} + \upsilon_i^T\mu_d \text{ for } i=1,...,N.
\end{aligned} \tag{4.52}$$

□

High-Confidence Safety Guarantee: Combining the uncertainty bound on d with the previous result encapsulated in Lemma 5 leads to the main result, summarized in the following Theorem.

Theorem 4. *Using the polytopic bounds* (4.49)*, the control action obtained from the quadratic program* (4.42) *guarantees robust safety (i.e. collision avoidance between agents) with probability at least* $1-\delta$.

Proof. Equation (4.49) can be represented in the form $\{Gd \leq g\}$; therefore, with probability $1-\delta$, the uncertainty d is contained in the set $\{d \in \mathbb{R}^n | Gd \leq g\}$ (by Lemma 6). From Lemma 5 and Equation (4.40), if one solves the quadratic program (4.42), it is guaranteed that $CBC(x,u,d) \geq k_c - H_1 d - u^T H_2 d - H_3 u \geq 0$ for all $d \in \{d \in \mathbb{R}^n | Gd \leq g\}$. Therefore, the CBF condition is satisifed with probability $1-\delta$, so safety is guaranteed with probability at least $1-\delta$ (by Definition 1 and the forward invariance property of CBFs [11]). □

4.5 Simulation Results: Multi-agent CBF

The algorithm is tested in a simulated multi-agent environment in which the robot, with nonlinear dynamics satisfying Assumption 2, navigates from a start to a goal position while avoiding collisions, in the presence of a random number of other agents (3-12 agents). Each of the other agents has a randomized (unknown) goal. Approximately half of these agents blindly travel from their start to the goal position without accounting for the movements of others, while the other half exhibit some collision avoidance behavior through their own control barrier functions (with random CBF parameters). An example simulation instance is shown in Figure 4.7. See the code (referenced below) for further simulation details/parameters and agent dynamics.

I simulate several instances of the other agents moving and interacting, and use this data for hyperparameter optimization of an MVG model as described in Section 4.4.3. The robot is then equipped with the robust CBF described in Section 4.4.2, using the optimized MVG for uncertainty prediction.

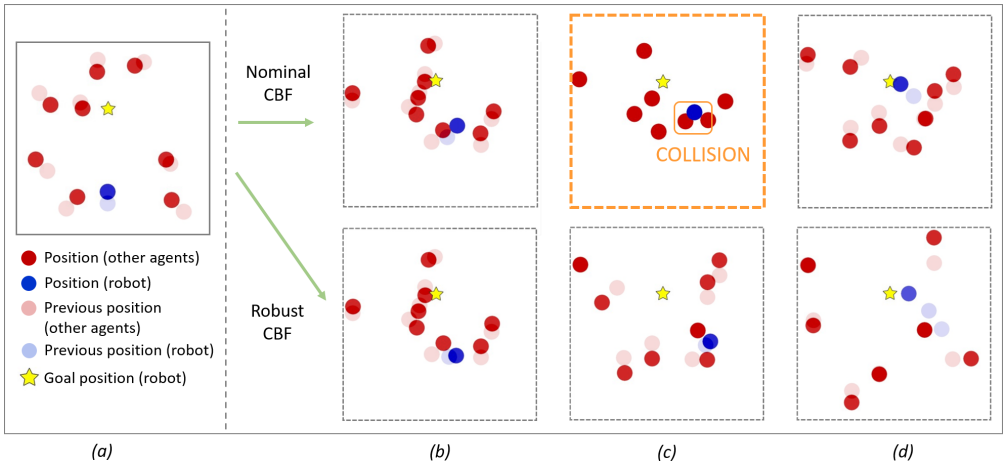


Figure 4.7: Sample path of a multi-agent system based on the nominal CBF (cf. [27]) and the proposed Robust CBF. The robot (blue) tries to navigate from a start position to random goal position while avoiding collisions with other agents (red). Approximately half of the other agents blindly travel towards their own randomly chosen goal, while the rest exhibit varying degrees of collision-avoidance behavior (the robot does not know their behavior a priori). *(a)* Initial robot/environment configuration, *(b)* Intermediate configuration, *(c)* Intermediate configuration showing that the nominal CBF controller experiences collision *(top)*, while the robust CBF avoids collision *(bottom)*. *(d)* Final configuration before robot reaches its goal position (star). See `https://youtu.be/hXg5kZO86Lw` for the simulation videos.

By running 1000 simulated tests in randomized environments, it is found that the robust CBF avoids collision in 98.5% of cases (when $\delta = 0.05$ [cf. (4.48)]),

performing much better than the nominal multi-agent CBF (cf. [27]), which avoids collisions in 85.0% of cases. The simulation results are summarized in Table 4.1.

	Robust Multi-Agent CBF	Nominal Multi-Agent CBF
Collision Rate	1.5 %	15.0%
Distance to Collision	7.4 ± 2.3	7.3 ± 2.1

Table 4.1: Performance statistics for the robust vs. nominal multi-agent CBF across 1000 randomized trials. For fair comparison, the robust and nominal CBFs were tested in the same randomized 1000 trials. **Collision Rate:** Percentage of trials that ended in collision. **Distance to Collision:** For trials without collision, the robot's margin from collision. The closer the robust CBF is to the nominal CBF, the less conservativeness is introduced by the uncertainty prediction.

Robustness must always come at the cost of performance (e.g. the robot can reach the goal faster if it does not care about collisions). However, the results in Table 4.1 show that the robust CBF only introduces slight conservativeness, as the margin from collision (in instances where the CBF was active) was very similar when utilizing the robust CBF vs. the nominal CBF.

The code for implementing the robust multi-agent CBF in the simulated environment can be found online [3]. A video of the simulations can be found at `https://youtu.be/hXg5kZO86Lw`.

4.6 Conclusion

This chapter examined how discrete-time CBFs could be effectively integrated with reinforcement learning to guarantee safety, while avoiding distortion of the policy gradient (thereby improving exploration efficiency). This chapter also showed how uncertainty in system/environment dynamics could be captured using GPs, and efficiently incorporated into a safe learning framework. Sections 4.1 and 4.2 looked specifically at polytopic CBFs with uncertainty modeled by multivariate GPs (dynamic uncertainty for each dimension is independent). Section 4.4 expanded this approach to the multiagent scenario with uncertainty modeled by MVGs.

The results in this chapter illustrate that CBFs are a promising tool for leveraging limited model information to guarantee safety under uncertainty in learning-based systems and guide RL exploration. Furthermore, their independence of the reward

function ensures safe operation regardless of any misspecification of the reward, which is known to be a significant issue in RL.

However, there is still much work to be done in expanding this methodology to more complex CBFs. This is not a huge issue when utilizing continuous-time CBFs, but discrete-time CBFs do not lend themselves as easily to QP formulations (which becomes an issue since almost all RL problems are defined with respect to discrete dynamics). Furthermore, while the proposed framework is not limited to considering uncertainty modeled by a GP – it can use any model approximation method that provides quantifiable uncertainty bounds (e.g. neural networks with dropout) – finding the right uncertainty model is an extremely challenging task. In particular, the assumption underlying GP models, discussed in Remark 3, is an extremely strong one. Therefore, providing more reliable safety guarantees for uncertain systems may necessitate moving away from GPs and discovering higher-fidelity uncertainty models, which will be discussed extensively in the next chapter.

Appendix C: Derivation of CBC lower bound (4.40)

Begin by expanding out the full CBC condition in (4.38), using the assumption of a relative degree 2 system. This leads to the following expression:

$$CBC(x,u,d) = \frac{(f_p(x)-f_p^h(x))^T(f_v(x)-f_v^h(x))}{\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|} + (\gamma-1)\frac{\Delta p^T \Delta v}{\|\Delta p\|}$$

$$\sqrt{a_{max}(\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|-D_s)} \; + \; (\gamma-1)\sqrt{a_{max}(\|\Delta p\|-D_s)} +$$

$$\Bigg[\frac{(f_p(x)-f_p^h(x))^T g_v(x)}{\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|}, \quad \frac{f_v(x)-f_v^h(x)}{\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|},$$

$$\frac{f_p(x)-f_p^h(x)}{\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|}, \quad -\frac{f_v(x)-f_v^h(x)}{\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|},$$

$$-\frac{f_p(x)-f_p^h(x)}{\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|} \Bigg] \times \Big[u_R \,, \; d_p \,, \; d_v \,, \; d_p^h \,, \; d_v^h \Big]^T +$$

$$u_R^T \Big[\frac{g_v(x)}{\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|} \,, \quad 0 \,,$$

$$\frac{-g_v(x)}{\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|} \,, \; 0 \Big] \times \Big[d_p \,, \; d_v \,, \; d_p^h \,, \; d_v^h \Big]^T +$$

$$\Big[0 \,, \; \frac{1}{\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|} \,, \; \frac{-1}{\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|} \,,$$

$$\frac{1}{\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|} \,, \; \frac{-1}{\|f_p(x)+d_p(x)-f_p^h(x)-d_p^h(x)\|} \Big] \times$$

$$\Big[u_R^T u_R \,, \; d_p^T d_v \,, \; d_p^T d_v^h \,, \; d_v^h d_p^h \,, \; d_v^T d_p^h \Big]^T. \tag{4.53}$$

By bounding the positional uncertainty terms $\|d_p(x)\| \leq \zeta_p(x)$ and $\|d_p^h(x)\| \leq$

$\zeta_p^h(x)$, a lower bound on $CBC(x,u,d)$ is obtained:

$$CBC(x,u,d) \geq \min\Big(\frac{(f_p(x)-f_p^h(x))^T(f_v(x)-f_v^h(x))}{\|f_p(x)-f_p^h(x)\| \pm \big(\zeta_p(x)+\zeta_p^h(x)\big)}\Big)+$$

$$\sqrt{a_{max}(\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)-D_s)} + (\gamma-1)\sqrt{a_{max}(\|\Delta p\|-D_s)}+$$

$$(\gamma-1)\frac{\Delta p^T\Delta v}{\|\Delta p\|} + \Big[\frac{(f_p(x)-f_p^h(x))^T g_v(x)}{\|f_p(x)-f_p^h(x)\|+\zeta_p(x)+\zeta_p^h(x)},$$

$$\frac{f_v(x)-f_v^h(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)}, \quad \frac{f_p(x)-f_p^h(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)},$$

$$-\frac{f_v(x)-f_v^h(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)}, \quad -\frac{f_p(x)-f_p^h(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)}\Big] *$$

$$\Big[u_R\ ,\ d_p\ ,\ d_v\ ,\ d_p^h\ ,\ d_v^h\Big]^T + u_R^T\Big[\frac{g_v(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)}\ ,\ 0\ ,$$

$$\frac{-g_v(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)}\ ,\ 0\Big] \times \Big[d_p\ ,\ d_v\ ,\ d_p^h\ ,\ d_v^h\Big]^T -$$

$$\frac{\zeta_p(x)\zeta_v(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)} - \frac{\zeta_p(x)\zeta_v^h(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)} -$$

$$\frac{\zeta_v^h(x)\zeta_p^h(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)} - \frac{\zeta_v(x)\zeta_p^h(x)}{\|f_p(x)-f_p^h(x)\|-\zeta_p(x)-\zeta_p^h(x)}. \tag{4.54}$$

Grouping terms, this can be rewritten as follows with the parameters defined in (4.41):

$$CBC(x,u,d) \geq k_c(x) - H_1(x)\mathbf{d} - \mathbf{u}^T H_2(x)\mathbf{d} - H_3(x)\mathbf{u}. \tag{4.55}$$

Chapter 5

SAFETY GUARANTEES UNDER LEARNED MODELS OF HUMAN BEHAVIOR

Chapter 4 addressed specifically the question of how to integrate a safety filter (CBF) into the reinforcement learning framework in order to guarantee safety under uncertainty, without distorting the learning process. Using learned uncertainty models, it is possible to provide probabilistic safety guarantees at some desired probability level, $1 - \delta$. However, it is crucial to note that all guarantees of safety rely on *accurate* models of the uncertainty. Indeed, all current research that utilizes machine learning to guarantee safety relies on the paradigm of

1. learning/modeling the uncertainty in the system/environment, and
2. designing a controller/planner that is robustly safe with respect to that uncertainty.

This chapter discusses the limitations of this paradigm and probabilistic uncertainty modeling in many safety-critical applications (Section 5.1), and then proposes an alternative approach to considering human uncertainty and ensuring safety when interacting with human agents (Section 5.2).

Before proceeding, let us consider what requirements might be necessary for safety-critical applications involving human interaction. In particular, if we want to obtain probabilistic safety guarantees for the system, what safety threshold δ would be acceptable?

Suppose a robot/car is guaranteed safe with probability δ across every $2s$ planning horizon. Given a safety threshold $\delta \approx 0.001$, one could expect a safety violation every hour; with a safety threshold $\delta \approx 10^{-6}$, one could expect a safety violation every 3 weeks. For reference, based on NHTSA data [127], human drivers would have an effective safety threshold of $\delta \leq 10^{-8}$ under this analysis.

While the choice of the safety threshold value, δ, is highly application-specific and subjective, based on the simple calculations above, it can readily be argued that

safety-critical robotic applications should strive for extremely low safety thresholds, on the order $\delta \leq 10^{-8}$ [149]. However, looking at the safety thresholds considered in the literature, as seen in Table 5.1, one finds that across different methods and different models of uncertainty, δ is almost always chosen such that $\delta \geq 0.001$. For many applications, this tolerance is wholly inadequate (e.g. would you ride in any autonomous vehicle that was guaranteed to have at most one collision per hour?). As discussed in the following section, the restriction to large δ arises due to fundamental limitations in data-driven modeling of uncertainty.

Uncertainty Model Class	Example Works	Min. Safety Threshold
Gaussian Process	[15, 37, 66, 86]	$\delta \geq 0.001$
Dynamics w/ Gaussian Noise	[68, 140, 181]	$\delta \geq 0.001$
Bayesian NN	[58, 93, 117]	$\delta \geq 0.05$
Noisy Rational Model	[69]	$\delta \geq 0.01$
Hidden Markov Model / Markov Chain	[111, 139]	$\delta \geq 0.01$
Quantile Regression	[57]	$\delta \geq 0.05$
Scenario Optimization	[32, 36, 144]	$\delta \geq 0.01$
Generative Models (e.g. GANs)	[81, 138, 143]	N/A

Table 5.1: Different model classes for capturing human trajectory uncertainty, used in previous safe planning algorithms in order to guarantee safety with probability $1 - \delta$. The right column shows the lowest safety threshold, δ, found used in the literature (in simulation or hardware experiments) for each model class. There is no entry for generative models, as these models have not yet been utilized to provide *explicit* safety guarantees during planning, though there is surely a trend in this direction.

5.1 Accuracy of Learned Models of Human Uncertainty

This section shows that the prevalent model classes of uncertainty (see Table 5.1) fail to accurately capture human behavior at safety-critical probability levels (which this work defines as $\delta \leq 10^{-8}$). These failures are revealed by testing prevalent modeling assumptions on real-world driving data from the highD driving dataset [100], which captures trajectory data from human-driven vehicles on German highways.

From the highD dataset, all trajectories of length 10 seconds are extracted, $\tau_{[0,10]}$, along with their corresponding environmental context, $\mathcal{E}_\tau$ (i.e. position/velocity of surrounding cars). The trajectories are then split into a training set, $(\tau^{(train)}_{[0,10]}, \mathcal{E}^{(train)}_\tau) \in \mathcal{D}^{train}$, and test set, $(\tau^{(test)}_{[0:10]}, \mathcal{E}^{(test)}_\tau) \in \mathcal{D}^{test}$. For every test trajectory, $(\tau^{(test)}_{[0:10]}, \mathcal{E}^{(test)}_\tau) \in$

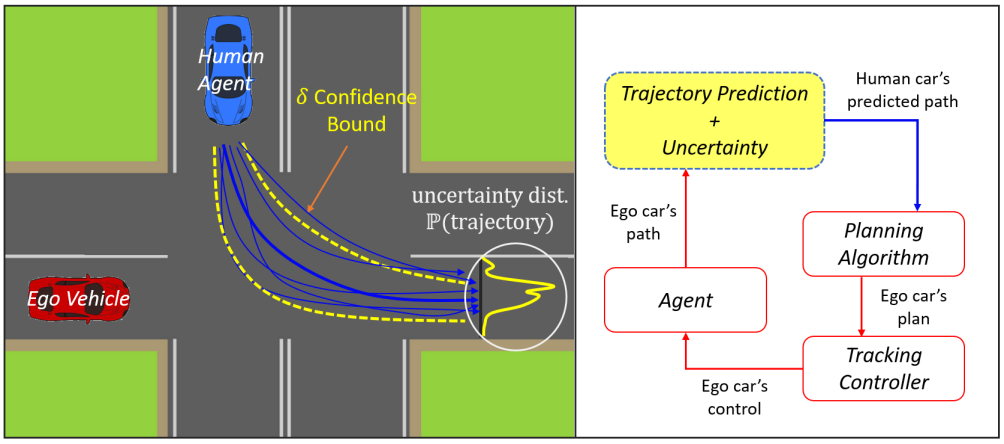


Figure 5.1: **(Left)** In this example, the red car must take into account the blue car's trajectory – and its uncertainty – in its plan to progress safely through the intersection. The dashed yellow curves denote the boundary of a tube that defines the δ confidence bound over trajectories. The white circle depicts a distribution over trajectories. The blue lines are example trajectories. **(Right)** Simplified illustration of different stages of the control pipeline. While every stage (prediction, planning, tracking) is crucial to guaranteeing safety, this paper focuses exclusively on the yellow box, *prediction*.

$\mathcal{D}^{test}$, all trajectories in the training set in equivalent scenarios are collected,

$$\mathcal{M}(\tau^{(test)}_{[0:10]}, \mathcal{E}^{(test)}_{\tau}) = \Big\{ \tau_{[0:10]} \;\Big|\; (\tau_{[0:10]}, \mathcal{E}_{\tau}) \in \mathcal{D}^{train} \,, \\ \|\tau_{[0:2]} - \tau^{(test)}_{[0:2]}\|_{\infty} < \epsilon \;\;,\;\; \|\mathcal{E}_{\tau} - \mathcal{E}^{(test)}_{\tau}\|_{\infty} < \epsilon_{\mathcal{E}} \Big\} \,. \tag{5.1}$$

Equivalent scenarios are defined as the set of trajectories with similar environmental context that are ϵ-close (ϵ = 2ft) over their first 2 seconds. Therefore, $\mathcal{M}(\tau^{(test)}, \mathcal{E}^{(test)}_{\tau})$ denotes the set of scenarios (within the training set, $\mathcal{D}^{train}$) that are equivalent to $(\tau^{(test)}, \mathcal{E}^{(test)}_{\tau})$. The choice of a past observation horizon of $2s$ follows [46], but the observed trends did not change considerably when using $1s$ or $3s$ for the past observation horizon.

For every test trajectory, $(\tau^{(test)}, \mathcal{E}^{(test)}_{\tau}) \in \mathcal{D}^{test}$, optimal parameters for an uncertainty model (e.g. Gaussian) are fit to the equivalent training scenarios, $\mathcal{M}(\tau^{(test)}, \mathcal{E}^{(test)}_{\tau})$, and observe where the test trajectory falls with respect to the computed distribution or bounds. By iterating through all trajectories in $\mathcal{D}^{test}$, it is possible to compute statistics analyzing how well the test trajectories fit to models predicted from the training trajectories.

Intuition in equations: To clarify this method and make clear the assumptions, let us now outline this approach in terms of different distribution errors. Let us define an agent's state $x = (\tau_{[0:2]}, \mathcal{E}_{\tau})$, and its action, a, as its future trajectory $a = \tau_{[2:10]}$.

Given that the future trajectory is drawn from some uncertain distribution, $a \sim \mathcal{A}(x)$, the goal is to learn a model $\hat{F}(x)$ that accurately approximates this distribution over trajectories, $\mathcal{A}(x)$, minimizing the following error,

$$L^{out} = \mathbb{E}\left[m\left(\mathcal{A}(x) \| \hat{F}(x)\right)\right], \tag{5.2}$$

where m defines some metric over probability distributions (e.g. total variation distance). The model $\hat{F}$ is trained on data from the training set $\mathcal{D}_{train}$. Since it is not possible to access to the true distribution, the expectation in (5.2) can be approximated using the test set, $\mathcal{D}_{test}$, yielding the error functions,

$$\begin{aligned} L^{unseen} &= \frac{1}{N_{unseen}} \sum_{x \in D^{test}} \left[m\left(\hat{\mathcal{A}}(x) \| \hat{F}(x)\right)\right], \\ L^{seen} &= \frac{1}{N_{seen}} \sum_{x \in D^{test} \cap \mathcal{D}^{train}} \left[m\left(\hat{\mathcal{A}}(x) \| \hat{F}(x)\right)\right], \end{aligned} \tag{5.3}$$

where $\hat{\mathcal{A}}(x)$ represents the approximation of $\mathcal{A}(x)$ based on $\mathcal{D}_{test}$, and N_{unseen}, N_{seen} are normalizing factors denoting the number of trajectories being considered. L^{unseen} can be interpreted as the test error, capturing how well the model $\hat{F}$ captures the action distribution $\hat{\mathcal{A}}$ from states it did not train on. On the other hand, L^{seen} captures how well $\hat{F}$ captures the action distribution, $\hat{A}$, from states it *has* trained on. In general, the relationship between these errors follows,

$$L^{out} \underbrace{\geq}_{\text{distribution gap}} L^{unseen} \underbrace{\geq}_{\text{generalization gap}} L^{seen}. \tag{5.4}$$

The analysis in this chapter focuses on L^{seen}. As this error ignores any generalization or distribution gap, it benchmarks the *best potential performance* of each model class. The distribution gap quantifies how the change from the true trajectory distribution to the test distribution $\mathcal{D}^{test}$ affects model accuracy. The generalization gap quantifies how out-of-distribution test examples affect the model accuracy.

Accounting for replanning: Most motion planning algorithms re-plan their trajectory at some fixed frequency (e.g. 1Hz). To account for this, prediction error (e.g. violation of the $\delta-$uncertainty bound) is examined only within a short re-planning horizon. I.e. the prediction must only be accurate within this replanning horizon. The horizon is set to 2 sec.

Incorporating conservative assumptions: To further highlight the fundamental limitations of learning uncertainty models of human behavior, since many prediction algorithms leverage goal inference, let us assume that an oracle provides the

target lane of every trajectory. Note that the aim is to illustrate limitations of learned probabilistic models, even under ideal conditions. Therefore, these strong assumptions (though unrealistic) help us reason about the best-case scenario for each model class, providing an upper-bound on performance.

Summarizing, this analysis of model uncertainty considers that: *(a)* there is no distribution gap, *(b)* there is no generalization gap, and *(c)* the target lane of every trajectory is given.

> *If the models perform poorly under these extremely generous assumptions, reasonable performance in realistic settings cannot be expected.*

Remark 9. The results in this section focus on in-distribution error (i.e. aleatoric uncertainty), rather than out-of-distribution error. In other words, this analysis highlights the fundamental inability of uncertainty models to accurately capture distributions at very low δ, regardless of generalization to unseen scenarios.

Remark 10. It is important to distinguish *motion predictors* from *uncertainty models*. While recent performance of motion predictors has drastically improved [81], they all leverage an underlying *uncertainty model* (see Table 1) to capture the probability of uncommon events. E.g. most neural network motion predictors output a Gaussian uncertain prediction. This section examines errors associated with *uncertainty models*, which propagate to the motion predictors.

5.1.1 Gaussian Uncertainty Models

Let us first analyze the popular Gaussian uncertainty model, used in most UQ neural networks [117], Gaussian process models [15], and robust regression [110, 126]. These approaches model the data and its uncertainty with a Gaussian distribution (see top 3 rows in Table 1).

Using the procedure outlined at the beginning of this section, I compute the best-fit Gaussian distribution, $\hat{F}$, over the training trajectories $\mathcal{D}^{train}$, and observe how well it captures the in-distribution test trajectories in $\mathcal{D}^{test}$ (i.e. minimizes L^{seen}). Figure 5.2 ($K = 1$) shows the ratio of observed to expected violations in the test set at each safety threshold, δ. A *violation* is defined when the test trajectory lies outside the δ-uncertainty bound (within a 2s re-planning horizon) for a specified δ. If the data followed a perfect Gaussian distribution, each curve in Figure 5.2 would track the dotted black line (i.e. keep a ratio near 1). However, the results show that while

the Gaussian model might be valid for $\delta \geq 0.01$, it is highly inaccurate outside this range, posing a problem for safety-critical applications.

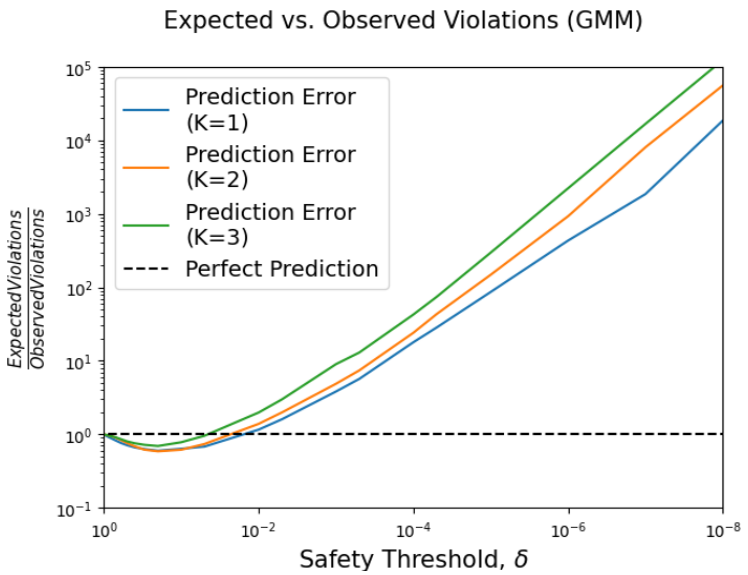


Figure 5.2: Prediction error vs. safety threshold, δ, using Gaussian mixture models on the highD dataset, considering a $2s$ re-planning horizon. K denotes the number of mixtures used, with $K = 1$ denoting a standard Gaussian distribution. The dashed black line represents a perfect prediction model.

Gaussian mixture models (GMM): One might point out that problems with the Gaussian model could be alleviated using GMMs over a discrete set of goals (e.g. left versus right turn). For example, interacting Gaussian processes (IGP) leverage this tool to alleviate the freezing robot problem [165]. However, when GMMs were trained on the same data with different numbers of mixtures ($K = 2, ..., 4$), prediction performance on test data did not improve for low δ (see Figure 5.2). These results illustrate limitations of any Gaussian-based uncertainty model (IGP, GMM, etc.), by highlighting that human behavioral variation is inherently non-Gaussian.

In addition to the issue of inaccurate distributional assumptions, the confidence bounds at level $\delta \approx 10^{-8}$ become very large, making planning around these bounds difficult or potentially infeasible. Figure 5.3 shows the 5σ confidence tube projecting the position of a car forward in time, based on the training data. Note that the 5σ ellipsoid (corresponding to $\delta < 10^{-6}$) encroaches on each lane, making it difficult for other cars to drive alongside it. This is because, although the car will typically stay in its lane, in rare instances (as shown in Figure 5.3) it will unexpectedly swerve into the other lane. This illustrates the difficulty of balancing the safety-efficiency tradeoff, as accounting for very rare events may be necessary for safety-critical applications, but this also introduces extreme conservatism.

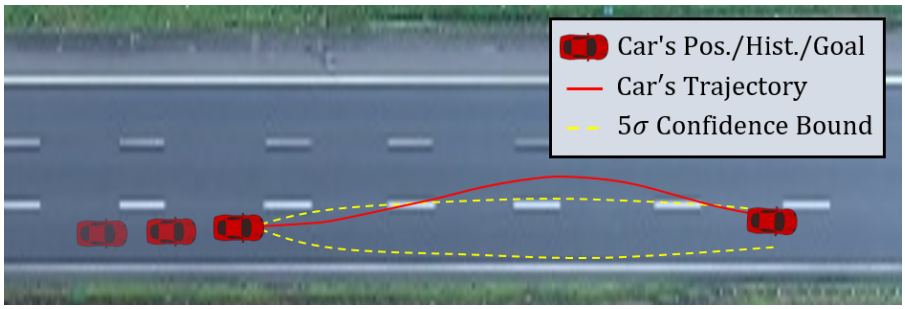


Figure 5.3: Example of a car's trajectory, along with the approximate 5σ confidence bound computed from the training trajectories, given the car's target lane 8 seconds in the future.

To further emphasize fragility of the Gaussian model at low δ, I generated synthetic 2D data from different, known distributions, and examined how well the best fit Gaussian predicted violations at a given δ. Even with perfectly i.i.d. training/test data, the error at low δ was significant. Details and results can be found in Appendix D at the end of this chapter.

5.1.2 Noisy Rational Model

The noisy rational model considers that humans behave approximately optimally with respect to some reward function. It has enabled significant progress in inverse reinforcement learning (IRL) by allowing researchers to learn reward functions from human data [141], and compute explicit uncertainty intervals over human agents' actions [69]. However, the noisy rational model adopts an underlying model of uncertainty in the exponential family, which places a strong assumption on the shape of the uncertainty distribution and assumes that there is a single "optimal" trajectory:

$$\mathbb{P}(x_{t+1} \mid \beta) = \frac{e^{\beta Q_H(x_{t+1})}}{\sum_{\tilde{x}_{t+1}} e^{\beta Q_H(\tilde{x}_{t+1})}}. \tag{5.5}$$

In the driving scenario, the optimal model simplifies to the Gaussian distribution, since $Q_H = \|x_{t+1} - \hat{x}_{t+1}\|_\Sigma$ for some Σ (i.e. the best fit to the data is sought). As a result, the issues illustrated in Figures 5.2 and 5.3 are exactly faced by the noisy rational model (i.e. the shape of the underlying distribution does not match the assumed distribution). Thus, even in the best case – known target lane, optimal data fit, no generalization gap – these models are ill-equipped to provide safety guarantees for safety-critical systems (e.g. $\delta < 10^{-8}$).

5.1.3 Quantile Regression

Quantile regression is an appealing alternative as it does not require strong assumptions on the underlying uncertainty distribution [57]. It is only concerned with

computing tubes such that $1-\delta$ proportion of trajectories are within that tube and δ are outside. To demonstrate its performance, quantile bounds are computed for each trajectory in the test set of the HighD dataset, using the equivalent scenarios from the training set. These quantile bounds are approximated as the smallest convex tube containing $1-\delta$ proportion of trajectories, which optimizes the expected mutual information between the state, x, and action, a [132].

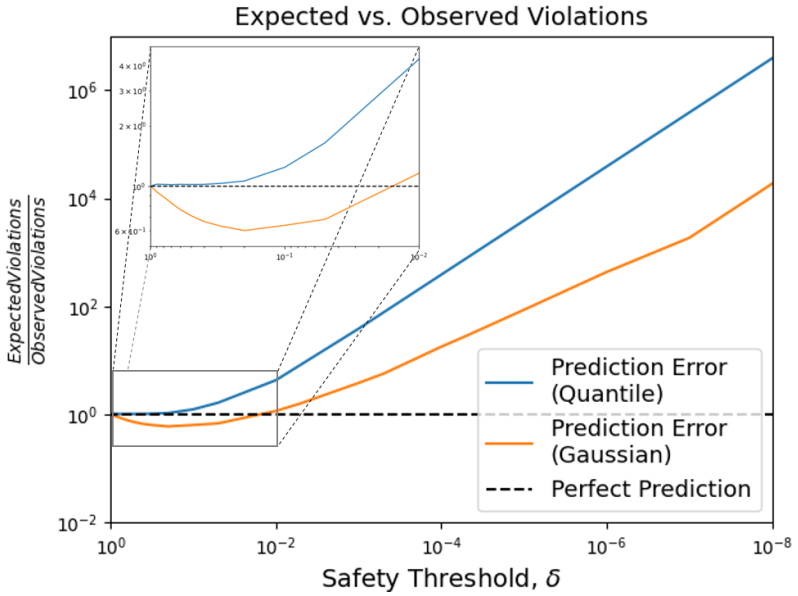


Figure 5.4: Prediction error vs. safety threshold, δ using computed quantile bounds or Gaussian uncertainty model on the highD dataset (assuming $2s$ re-planning horizon). The dashed black line represents a perfect prediction model.

Figure 5.4 shows the ratio of the observed to expected number of test trajectories outside each quantile at safety threshold δ. As seen in the plot, the quantile regression model performs much better than the Gaussian model for $\delta > 0.1$. However, performance rapidly deteriorates as δ decreases, making estimated confidence bounds meaningless, since they fail to predict violation probabilities.

This result makes sense, as obtaining accurate quantile bounds at the δ-confidence level relies on splitting the data: δ percent of points should be outside the quantile bound with the rest inside those bounds. However, little (if any) data is available outside the quantile bound for very low δ. Put differently, to *observe* a one-in-a-million event, at least a million trajectories must be observed. To *reliably predict* those events, many more trajectories are needed.

Improving Accuracy with Increasing Data: Given the availability of increasingly large robotics datasets, one should ask if it is possible to reach good accuracy at

desired safety thresholds, δ, by using more data. To answer this, define the smallest accurate safety threshold, δ_{min}, as follows,

$$\delta_{min} \;=\; \min\ \delta \quad \text{such that} \quad \left|\log\left(\frac{\text{expected}(\delta)}{\text{observed}(\delta)}\right)\right| \leq \varepsilon\ . \tag{5.6}$$

Here, $\varepsilon = 0.5$, where ε represents the vertical distance between each curve in Figure 5.4 and the dotted black line. Thus, δ_{min} represents the smallest δ such that the computed quantile bounds are ε-accurate. Note that δ_{min} is computed with respect to a given set of data. Therefore, by varying the size of the training set, one can capture how δ_{min} varies with the amount of training data, shown in Figure 5.5a. The trend shown in Figure 5.5a is surprisingly linear ($r^2 = 0.995$), which held across different sections of the dataset (i.e. different highways). This scaling is consistent with the lower bound on sample complexity derived in [52], shown in Equation (5.7) and discussed further below.

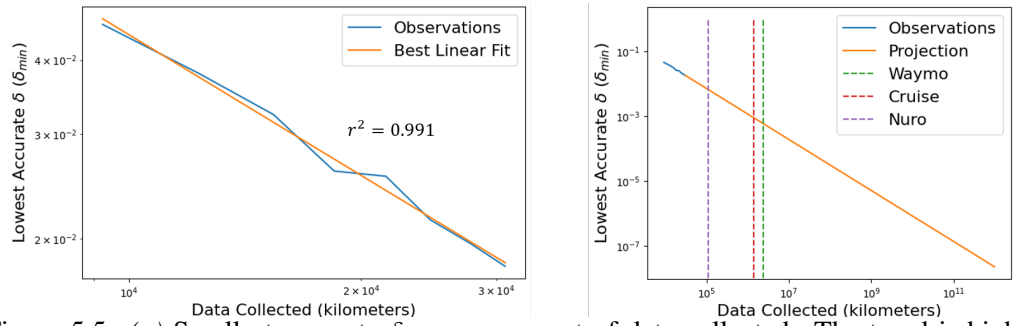


Figure 5.5: *(a)* Smallest accurate δ versus amount of data collected. The trend is highly linear ($r^2 \approx 0.995$), and holds across different sections of the dataset. *(b)* Projection showing the expected amount of data required to obtain an accurate safety threshold δ_{min}. The dashed lines show the number of kilometers driven in California in 2019 by Waymo, Cruise, and Nuro.

While initially promising, if this linear trend is extended down to $\delta_{min} \approx 10^{-8}$, it can be seen that the amount of data required to reach safety-critical thresholds is far from feasible. Figure 5.5b shows that trillions of kilometers of driving data are needed to achieve accurate quantile bounds, even under extremely generous assumptions (e.g. perfect generalization). For reference, in 2018, approximately 5 trillion kilometers were driven in total across *all* cars/trucks in the U.S. [127].

The same analysis was conducted on synthetic 2D data, and found the same trends seen in Figures 5.4 and 5.5. Details and results can be found in Appendix E at the end of this chapter.

Quantile Regression as a Fundamental Limitation: One might be tempted to conclude from Figure 5.5b that we should look for alternative methods (to quantile regression) that have better data efficiency / sample complexity. Note that all methods providing confidence bounds over trajectories at a specified safety threshold δ can be fundamentally viewed as classification problems; one must classify $1-\delta$ trajectories within some learned bounds, with the rest outside those bounds. By viewing this as a classification problem, it is possible to leverage results from VC-analysis that lower bound the data required, N, to reach a given prediction confidence [52]: to guarantee $Pr(\text{error}) \leq \delta$, it is necessary that [1]

$$N(\delta, M) = \Omega\Big(\frac{1}{\delta}\ln\big(\frac{1}{\delta}\big) + \frac{\text{VCdim}(M)}{\delta}\Big), \tag{5.7}$$

where VCdim(M) is the VC dimension of the utilized model M (see [52] for proof). The linear trend in Figure 5.5 (showing $N(\delta) \propto \frac{1}{\delta_{min}}$) fits very nicely with the lower bound (5.7), given that the second term dominates the first (i.e. the VC dimension is large). Note that if the first term dominated the second term, *worse* data scaling would be expected.

This analysis suggests that alternative methods cannot provide confidence bounds with better data scaling than shown in Figure 5.5.

5.1.4 Scenario Optimization Model

Scenario Optimization is an appealing approach because (like quantile regression) it does not assume an underlying distribution over the data [144]. It relies only on the assumption that the data is drawn i.i.d. from some fixed (unknown) distribution. Therefore, one can obtain a high-confidence bound on the probability that a new trajectory is inside or outside a computed tube, *without strong assumptions on the underlying distribution*.

With this approach, the safety threshold, δ, is a direct function of the amount of observed data [30]; in other words $\delta = \delta(N)$, where N is the number of training trajectories or "samples." Therefore, arbitrarily small confidence levels (e.g. $\delta = 10^{-8}$) cannot be chosen. While this prevents users from applying the approach inappropriately, it requires very large amounts of data to yield low enough confidence levels for safety-critical applications. For example, with 40,000 trajectories, it is

[1] The argument of Big-Omega Ω here represents a lower bound on the asymptotic sample complexity.

possible to approach $\delta \approx 10^{-4}$ (after this point, computational feasibility became an issue). This suggests it is not feasible to reach desired δ levels given realistic datasets.

Using the highD dataset and treating the trajectories in the training set as observed samples, high-confidence bounds were obtained (computed as the convex hull of the training trajectories) such that new trajectories should lie within those bounds with probability at least $1 - \delta$. For example, Figure 5.6 shows the predicted confidence bounds for an example driving instance; the scenario optimization approach predicts that the future trajectory of each car should fall within the blue confidence bounds at $2, 4, 6$ seconds in the future with 97.1% probability.

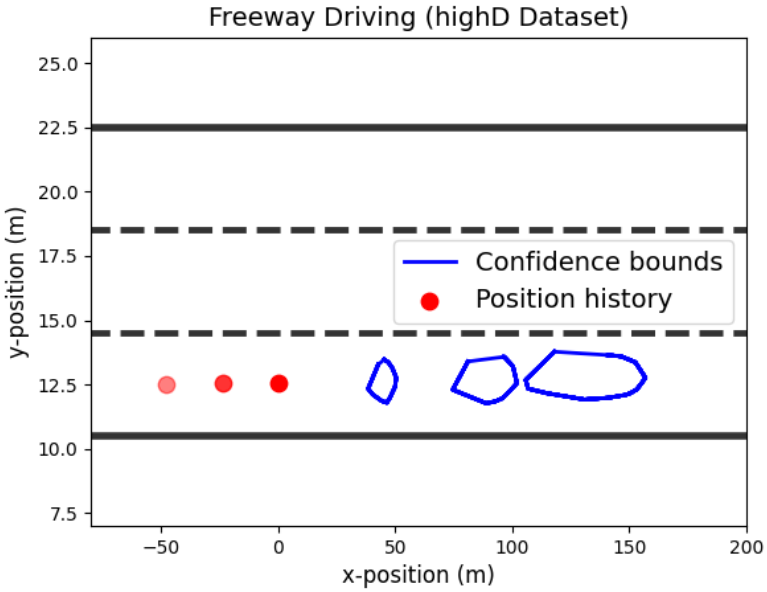


Figure 5.6: Plot of confidence bounds over the car's future trajectory, where the car's target lane is known. The car's positional history is shown by the red circles, and training data is taken from equivalent scenarios in the highD dataset. Calculations show that there is a 97.1% probability that a new trajectory falls within the blue confidence bounds at 2, 4, and 6 seconds in the future.

To test the accuracy of the computed confidence bounds, I examined how often trajectories in the test set actually remained within those bounds in the highD dataset (over a 2 second horizon). The ratio of observed violations to expected violations was smaller than expected (i.e. the method was conservative), which is reassuring for safety. Specifically, the observed vs. expected percentage of violations was approximately 4% vs. 15%.

However, the safety threshold $\delta(N)$ was always large ($\delta \in [0.02, 0.44]$) and unable

to be arbitrarily defined, which makes the scenario optimization approach currently inapplicable to many safety-critical applications. This is consistent with the conclusion in the *quantile regression* section that much more data is necessary to obtain reliable, probabilistic bounds.

5.1.5 Hidden Markov Models

Rather than reasoning about uncertainty only over trajectories, many methods in the POMDP literature reason about uncertainty over discrete intentions. Most often, these discrete intentions denote different goal positions for the agent, but they could also denote different operational modes (e.g. yield vs. no yield). Hidden Markov models enable an agent's most likely intention to be computed, which proves useful in solving many challenging problems. However, when guaranteeing safety with safety threshold δ, intention must be correctly inferred with probability $1-\delta$. Issues arise when the intention must be inferred with very high confidence $\delta \leq 10^{-8}$.

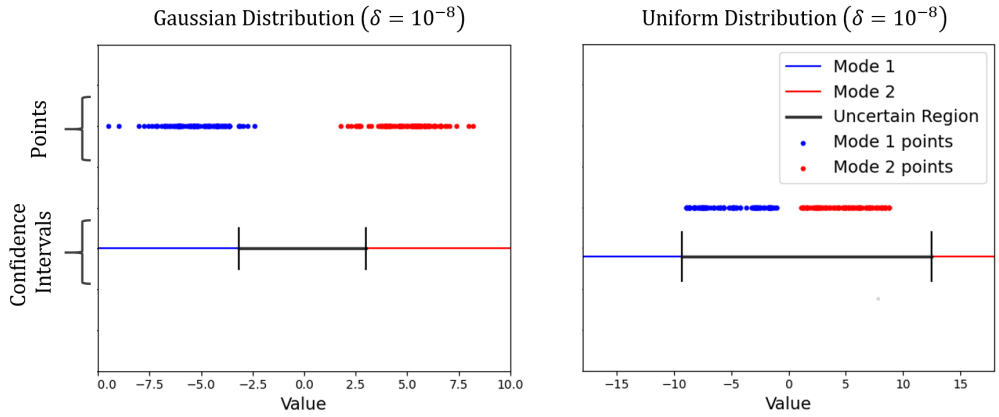


Figure 5.7: Synthetic data is generated from two different modes: *(mode 1 – blue, mode 2 – red)*. The confidence intervals below denote where a point would have to lie in order to classify it, with confidence $\delta = 10^{-8}$, as coming from either mode 1 or mode 2. For example, if a new point falls in the interval covered by the blue bar, it can be classified as coming from mode 1 with confidence $\delta \leq 10^{-8}$. If it falls anywhere in the gray interval, its mode cannot be deduced (assuming a uniform prior).

This brief analysis is illustrated with a 1D toy problem using synthetic data. I generated 1000 i.i.d. data points from two distinct distributions (mode 1 and mode 2), and computed the best fit Gaussian for each of these distributions. Note that the results did not change when increasing the amount of data. I then computed the intervals in which a new point would have to lie in order for us to classify it in either mode 1 or mode 2 with $1-\delta$ confidence. This was done by applying Bayes rule,

assuming a uniform prior over the modes,

$$\mathbb{P}(\text{mode} \mid x) = \frac{\mathbb{P}(x \mid \text{mode})\ \mathbb{P}(\text{mode})}{\mathbb{P}(x)}. \tag{5.8}$$

Figure 5.7 shows these intervals when the points were generated from either a Gaussian distribution or a uniform distribution. The interval covered by the gray line denotes the interval in which one can *not* classify (with δ confidence) a point's mode. Note that the gray line extends across a significant portion of the data range, but is reasonable when the underlying distribution of points in each mode is *perfectly Gaussian*. However, when the generated data is uniformly random, the uncertainty interval stretches across the entire range of data. This suggests that inferring intention or hidden "modes" under uncertainty will often be infeasible when considering very low safety thresholds, δ, especially since it was shown above that human behavioral variation is non-Gaussian. Furthermore, it is not possible to compensate for this non-Gaussian variation as accurate knowledge of the true underlying distribution is not available.

5.1.6 Generative Models

Generative models have garnered significant interest in trajectory prediction because of their ability to *implicitly* learn the distribution $\mathcal{A}(x) = p(a|x)$. However, there are two significant issues with these approaches, the first of which is the time required to utilize these models in safety-critical situations. For example at best, a single prediction takes $\approx 0.05s$ with Social-GAN [81], or $\approx 0.001s$ with Trajectron [143]. In order to guarantee safety with probability $\delta = 0.01$, it would be necessary to generate 100 trajectories taking $> 0.1s$. To guarantee safety with probability $\delta = 10^{-8}$, 10^8 trajectories must be generated, taking $> 100,000s$ (> 1 day), which is not suitable for real-time operation. While computational cost will surely decrease over time, it is unclear whether this modeling approach will be feasible in the near future.

However, more importantly, there are no guarantees that the uncertainty distribution implicitly captured by generative models provides any reasonable approximation to the true uncertainty distribution. It has been shown – both empirically and theoretically – that GANs can fail to learn the true distribution (suffering from mode collapse), even when their training objective nears optimality [16]. Furthermore, the theoretical data efficiency bound described by Equation (5.7) suggests that the implicit distribution learned by such models will be inaccurate (at the safety thresholds being considered) without currently infeasible amounts of data.

Note on Uncertainty Qualification (UQ) Neural Networks: UQ neural networks have not been discussed because neural networks do not compose a distinct class of uncertainty models. Instead, they only provide a functional representation of the uncertainty in a given class (e.g. UQ neural networks typically output a Gaussian distribution [56]). The results provided above highlight *best-case* performance bounds for each class of model uncertainties, given optimal fit to the data. Thus, using neural networks to parameterize the model uncertainty will only yield worse performance.

5.1.7 Section Summary

The main message of this section is that *even under extremely generous assumptions*, current models of human uncertainty are unable to extend safety guarantees to the confidence levels, e.g. $\delta < 10^{-8}$, that are needed for widespread adoption of safety-critical autonomy in human environments.

Learned uncertainty distributions become highly inaccurate at low δ, undermining any claimed guarantees of safety.

There is a fundamental limitation to modeling human uncertainty purely as a random process. Data-driven methods (i.e. machine learning) are designed to capture prominent patterns in data and predict *likely* events; they are not suited to predict rare events. Intuitively, a million samples are needed to *observe* a one-in-a-million event, and many more samples are required to *reliably predict* those events. While it is theoretically possible that huge datasets could eventually enable accurate prediction of rare events, this analysis shows that such amounts of data are far from feasible in the near future (even ignoring generalization issues and computational cost).

Human uncertainty vs. sensor-based uncertainties: Even if a system must be certified safe with $\delta = 10^{-8}$, it is uncommon to require any single module to have a failure probability less than 10^{-8}. Instead, redundancy with multiple, independent modules can help certify system safety. For example, in a robotic system, two different modules (one using LIDAR and one using stereo cameras) might predict an obstacle's current position, each with confidence $1 - 10^{-4}$. Then the overall system can reason about the obstacle's position with confidence $1 - 10^{-8}$. The key is that *the modules must be independent*. While this may be a fair assumption for sensing uncertainty, it is not fair for human behavior prediction. However, this could be a promising avenue of research: to simultaneously learn multiple predictors [61], *while enforcing independence between their predictions*.

5.2 Guaranteeing Safety among Humans through Behavioral Contracts

From the analysis in the previous section, one might be tempted to draw the conclusion that either much more data must be gathered or that better parameterizations for human behavioral uncertainty must be devised. However, this section advocates instead for a shift away from modeling trajectory uncertainty as a purely random process, arguing that assumptions on human intention must be intelligently incorporated into uncertainty prediction. I propose that *assume-guarantee contracts* with learned components be used as a formal framework for guaranteeing safety of robots interacting with humans, which will allow human uncertainty to be naturally bounded by considering human intentions, and imposing some responsibility upon human agents. Indeed, if human uncertainty is treated entirely as a random process, then extremely rare events must always be accounted for, making interactive navigation in many scenarios impossible [166].

Let us consider a simple thought exercise to illustrate the necessity of an assume-guarantee contract framework. Consider an agent that interacts with a single other agent and has 3 abstract actions it can take, with different rewards for each action. However, if both agents take the same action (e.g. collision), they each receive a large penalty. This scenario is depicted in Figure 5.8 (where the illustration on the left provides a concrete example of what these actions could be).

If a distribution over the actions of the other agent is learned from data, one would find that there is a non-trivial probability that it will take either action a_2 or a_3, meaning action a_1 must be taken in order to guarantee safety with probability $> 80\%$. However, this is clearly non-sensical – just because we have observed other agents in the past take an aggressive left turn does not mean we must plan assuming the other agent will do so!

Even if it were possible to estimate the other agent's reward function, one would find that both actions a_2 and a_3 belong to valid nash equilibria, meaning the only "safe" action for the main agent to respond with would again be a_1. There is a rich literature built on inverse reinforcement learning (see Section 1.3.4) dealing with learning and leveraging other agents' reward functions to interact with them. However, uncertainty modeling is very difficult with these approaches (in addition to accurate reward learning).

The fundamental issue is that no responsibility is placed on the other agent in this interactive navigation scenario – since their uncertainty is treated as a random variable, the other agent may take any available action as long as it passes a certain

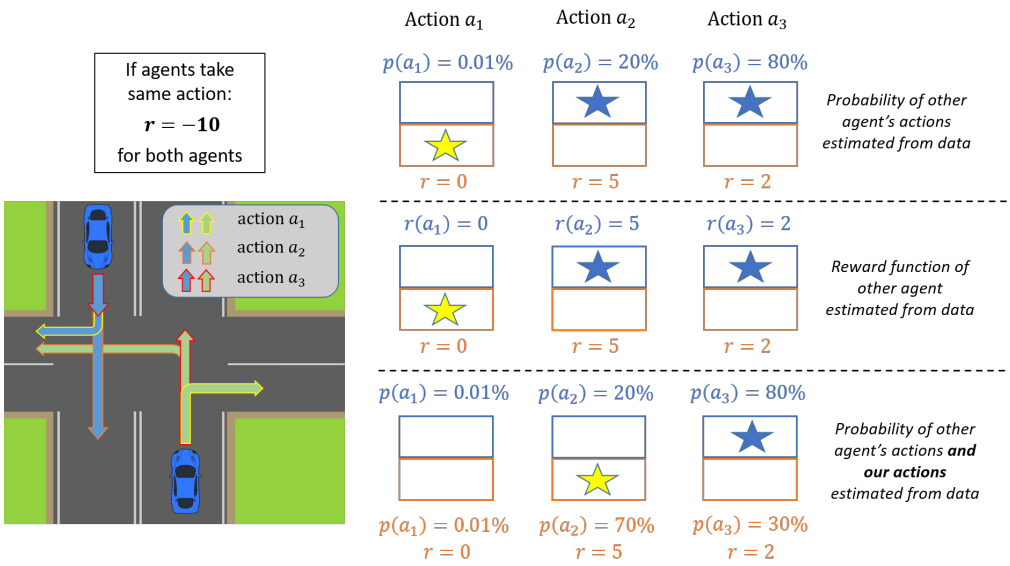


Figure 5.8: Toy example illustrating that treating uncertainty in human actions can lead to unintuitive, inefficient plans. The image on the left shows a simplified example of two cars that must safely interact when given 3 potential actions. If they both choose the same action, a collision will occur. On the right, different scenarios on what actions could be safely taken are outlined based on how the "other" agent's action is predicted (the "other" agent's potential actions are denoted by the blue stars and "our" agent's safe action is denoted by the yellow star).

threshold of probability (even if that action is a one-in-a-million event). In reality, humans agents are non-random; their uncertainty is bounded by social rules and obligations.

The big question then is: how should these rules and obligations be encoded? For example, some works assume that other agents are always incentivized to avoid collisions when possible [165]. Another approach is to consider that the reward function of other agents can be learned, and therefore their actions can be backed out by solving for an optimal solution of some game [69, 141]. Other works show that safety can be guaranteed if all agents obey strict rules on the ordering and coordination of decisions [134]. However, current approaches impose overly strict assumptions (e.g. agents always avoid collision, follow preset rules, obey learned reward function), are uninterpretable, and/or utilize inaccurate models of uncertainty. This section argues that assume-guarantee contracts with learned subcomponents provide an interpretable way to encode rules and obligations that are not overly strict, helping agents to balance safety and efficiency.

5.2.1 Problem Setup:

Consider a multi-agent environment with agents $Ag = X, 1, ..., M$, where X indexes the controlled agent. We want to plan a safe, efficient trajectory for agent Ag_X from its current position to a goal position in the presence of M other unknown agents $(Ag_1, ..., Ag_M)$. Suppose that perfect observations of the environment, $o = (o_X, o_1, ..., o_M)$, are available, where o_i denotes the position/velocity/acceleration of agent i. Let $a = (a_X, a_1, ..., a_M)$ denote the action of all agents, such that a_i denotes the position of agent i over time horizon T (i.e. $a_i = (p_i^{t+1}, p_i^{t+2}, ..., p_i^{t+T})$).

Define $\mathcal{A}_X(o_X)$ and $\mathcal{A}_i(o_i)$ as the set of all feasible actions for agent X and agent i, respectively, given the observations o_X, o_i. The goal is to infer accurate restrictions on the other agents' actions $\mathcal{W}_i(o) \subseteq \mathcal{A}_i(o_i)$, such that it is possible to plan *safe and efficient* actions for agent $a_X \in \mathcal{A}_X(o_X)$ by avoiding $\mathcal{W}_i(o)$. Define a binary collision-checking function $coll : \mathcal{A}_X \times \mathcal{A}_i \rightarrow \{0, 1\}$, such that safety is violated between agent X and i if and only if $coll(a_X, a_i) = 1$.

Note that the aim is *not* to accurately predict the behavior of the other agents (i.e. infer $\mathcal{W}_i$), but rather it is only to ensure that predictions enable safe, efficient interactions. For example, in the scenario depicted in Figure 5.8, we do not care if the other agent took action a_1 (instead of the predicted action a_3) as long as it did not conflict with action a_2 of agent X.

5.2.2 Preliminaries

Agent Signaled Intentions: *Signaled intentions*, $\mathcal{Q}$, dictate the interaction between two agents. Every agent j has a signaled intention with respect to every other agent i at each time step t, $q_{i,j}^{(t)} \in \mathcal{Q}$, which constrains the set of actions agent j can take. This work considers the set of intentions $\mathcal{Q} = \{$ `distracted`, `yielding`, `aggressive`, `adversarial` $\}$, which describes the way in which one agent intends to interact with another (though this set of signaled intentions can be much more general). Since this signaled intention is not explicitly communicated, let $\hat{\mathcal{Q}}_{i,j}^{(t)}$ denote agent i's belief over $q_{i,j}^{(t)}$ (i.e. the set of signaled intentions agent j might have with respect to agent i at time t). For example, we might have $q_{i,j}^{(t)} =$ `distracted`, but $\hat{\mathcal{Q}}_{i,j}^{(t)} = \{$`distracted`, `aggressive`$\}$ since agent i is uncertain about $q_{i,j}^{(t)}$.

Options: Options $\mathcal{Y} \subseteq \mathcal{P}(\mathcal{A})$ denote learned higher-level sets of actions (i.e. macro-actions) that correspond to agents' different signaled intentions. These options are learned based on relationships between (1) agents' expected actions $\mathcal{A}_i^{(exp)}$, (2) agents' reasonable action $\mathcal{A}_i^{(reas)}$, and (3) agents' feasible action $\mathcal{A}_i^{(feas)}$. The

Signaled Intention $q_{i,j}$: Agent j's signaled intention for interaction with agent i	**Obligations** Mapping from signaled intention to allowable actions	**Options (Data-Driven)** Learned sets of actions that form the basis of obligations
$q_{i,j}$ = distracted	Take any action that can be expected	$\{ a_j \in \mathcal{A}^{exp}(s_j) \}$
$q_{i,j}$ = yielding	Take *reasonable, unexpected* actions that lead further from collision	$\left\{ \begin{array}{c} a_j \in \mathcal{A}^{reas}(s_j) \setminus \mathcal{A}^{exp}(s_j) \quad \mid \\ d\left(a_j, \mathcal{A}^{exp}(s_i)\right) > d\left(\mathcal{A}^{exp}(s_j), \mathcal{A}^{exp}(s_i)\right) \end{array} \right\}$
$q_{i,j}$ = aggressive	Take any *reasonable* actions	$\{ a_j \in \mathcal{A}^{reas}(s_j) \}$
$q_{i,j}$ = adversarial	Take any *feasible* action	$\{ a_j \in \mathcal{A}^{feas}(s_j) \}$

Figure 5.9: Overview of the signaled intentions considered in this work. The obligations map these signaled intentions to a specific option (set of actions), described in each row.

limited set of options used in this work are defined below, though many others could be defined for more refined restrictions.

$$
\begin{aligned}
y_i^{dist}(o) &:= \mathcal{A}^{exp}(o_i) \\
y_i^{aggr}(o) &:= \mathcal{A}^{reas}(o_i) \\
y_i^{adv}(o) &:= \mathcal{A}^{feas}(o_i) \\
y_i^{yield}(o) &:= \{a \in \mathcal{A}^{reas}(o_i) \mid d\left(o_i^t, \mathcal{A}^{exp}(o_X^{t-1})\right) + \epsilon_{yield} < \\
&\qquad\qquad d\left(\mathcal{A}^{exp}(o_i^{t-1}), \mathcal{A}^{exp}(o_X^{t-1})\right)\}.
\end{aligned}
\tag{5.9}
$$

Obligations: Obligations, $\mathcal{B} : Q \times O \rightarrow \mathcal{Y}$, map signaled intentions and observations to options (i.e. allowable sets of actions). For example, an agent that is distracted has a smaller set of allowable actions (i.e. more restrictive option) than an agent that is aggressive (see Figure 5.9). Inference of these obligations will enable safe, efficient planning by constraining the sets of actions that agents around us can take.

The relation between signaled intention, options, and obligations is summarized in Figure 5.9. All of these components (signaled intentions, options, obligations) make up an ordered contract automaton (defined below), which will form the backbone of the proposed modeling framework for capturing multiagent interaction. Figure 5.10 provides an overview of the proposed approach to safe multi-agent interaction.

Definition 3. An ordered contract automaton is a tuple $(Q, \mathcal{B}, \mathcal{Y}, \mathcal{A}, T_q, \preceq, Q^{(0)})$, where

- Q is the set of possible signaled intentions;

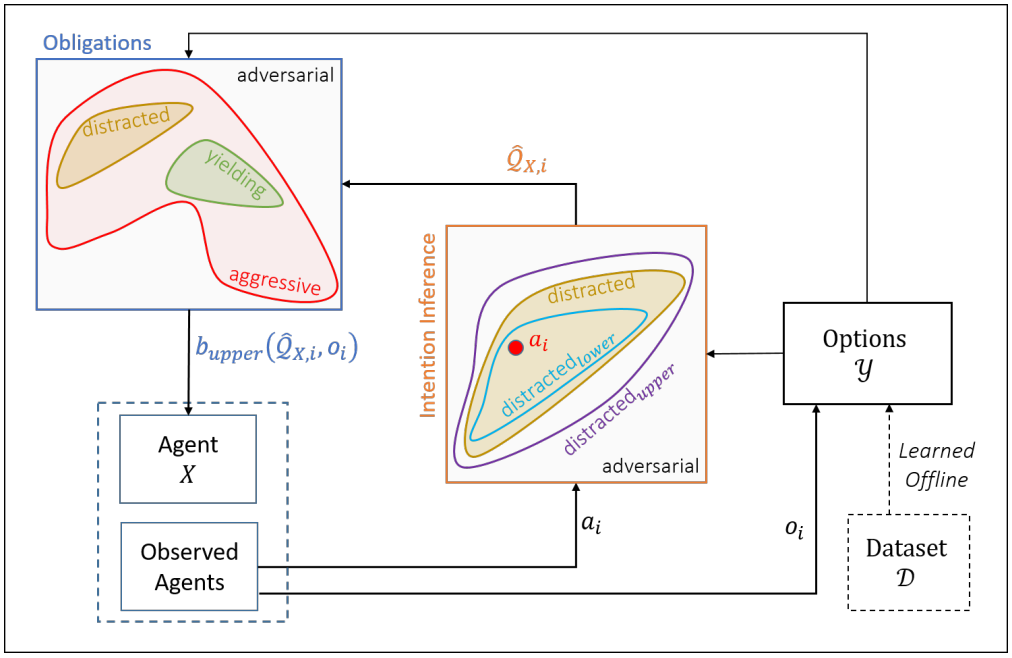


Figure 5.10: Diagram depicting the overall framework for the proposed safe interactive planning algorithm.

- $Q^{(0)}$ is the set of initial signaled intentions an agent could have;
- $\mathcal{A}$ represents the set of agent actions;
- $\mathcal{Y} \subseteq \mathcal{P}(\mathcal{A})$ represents the set of agent options;
- $\mathcal{B}$ is the set of *obligations*, $b : Q \times O \to \mathcal{Y}$, defining constraints over agents' actions given signaled intention. An action $a \in \mathcal{A}$ satisfies obligation b under signaled intention q if and only if $a \in b(q, o)$;
- T_q is a deterministic transition function $T_q : Q \times \preceq \times \mathcal{P}(\mathcal{Y}) \to Q$;
- $\preceq$ is a partial order over options $\mathcal{Y}$.

It is required that for all $y \in \mathcal{Y}$:

$$\forall (q, y_1, q_1), (q, y_2, q_2) \in T_q : y_1 \neq y_2 \;\to\; (y_1 \prec y_2 \,||\, y_1 \succ y_2). \tag{5.10}$$

This condition ensures that there is a unique outgoing transition for every action, which is determined based on the partial ordering $\preceq$.

5.2.3 Behavioral Contracts

The core of the safety guarantees provided by the proposed framework stem from behavioral contracts. They allow us to systematically and interpretably divide responsibility between agents for the actions they are allowed to take when interacting with others. When all agents satisfy their behavioral contracts, safety is guaranteed. When they do not, the contracts allow us to identify *blameworthy actions* [134] that point to why an accident occurred and who was responsible, providing interpretability.

Behavioral Contracts: A behavioral contract is defined as a special case of the more general assume-guarantee (A/G) contract, which is composed of *(a)* a set of assumption variables, *(b)* a set of guarantee variables, *(c)* a relation between the assumptions and guarantees. Such a contract, $(\mathbb{A}, \mathbb{G})$ is defined by $\mathbb{A}$, specifications over the assumption variables, and $\mathbb{G}$, specifications over the guarantee variables. Behavioral contracts are defined as follows:

Definition 4. A behavioral contract between agent i and another agent j is a 2-tuple $(\mathbb{A}_{i,j}, \mathbb{G}_{i,j})$ where,

- $\mathbb{A}_{i,j}$: The assumption that agent i makes regarding agent j is that *either* (1) $a_j \in b(q_{i,j}^{(t)}, o_j)$ or (2) $coll(a_j, y_i^{aggr}) = 0$ and $a_j \in b(q_{i,j}^{t+1}, o_j)$.
- $\mathbb{G}_{i,j}$: The guarantee that agent i makes to agent j is that $coll(a_i, b(q_{i,j}, o_j)) = 0$ and *either* (1) $a_i \in b(q_{j,i}, o_i)$ or (2) $coll(a_i, y_j^{aggr}) = 0$ and $a_i \in b(q_{j,i}^{t+1}, o_i)$.

The contract $(\mathbb{A}_{i,j}, \mathbb{G}_{i,j})$ is satisfied (i.e. $(a_j, a_i) \vDash (\mathbb{A}_{i,j}, \mathbb{G}_{i,j})$) if and only if $a_j \vDash \mathbb{A}$ and $a_i \vDash \mathbb{G}$. Otherwise, either agent i or agent j (or both) must have violated their end of the contract. I provide a more intuitive translation of the assumptions and guarantees of each agent below:

- Assumption $\mathbb{A}$: If agent j's action violates its obligation, this action must reflect its new signaled intention $q_{j,i}^{(t+1)}$ and avoid collision with any of agent i's reasonable actions.
- Guarantee $\mathbb{G}$: Agent i guarantees that both,
  - Agent i's action avoids collision with agent j's obligation,
  - If agent i's action violates its obligation, this action must reflect its *new* signaled intention $q_{i,j}^{t+1}$ and avoid collision with any of agent j's reasonable actions.

Our approach to motion planning for a given agent Ag_X is to (a) estimate each agent's signaled intention with respect to Ag_X, (b) determine the corresponding obligations $b(q_{X,i}, o_i)$ that each agent has to Ag_X, (c) synthesize a trajectory for Ag_X that is safe with respect to the obligations of all other agents $b(q_{X,i}, o_i)$, and accurately reflects its signaled intention $q_{i,X}$. Taking this approach, Section 5.2.4 shows that the safety of the resulting multiagent system is guaranteed under these behavioral contracts.

Remark 11. It is NOT necessary that each agent always satisfy its obligations ($a_i \in b(q_{X,i}, o_i)$). In fact, agents are expected to frequently violate those obligations. Instead, it is necessary that obligation violations accurately reflect new signaled intentions.

The missing pieces then are: (1) given an agent's intentions, how are its obligations/options learned from data (i.e. learn the mapping $b(q_{i,j}, o_i)$), and (2) how are agents' intentions $q_{i,j}$ accurately inferred?

Learning Bounds on Obligations

Let us suppose that at every time step, agent Ag_X has a conservative belief over agent i's signaled intention, $\hat{Q}^{(t)}_{X,i}$, such that $q_{X,i} \in \hat{Q}^{(t)}_{X,i}$. Based on the defined obligations, this conservative belief allows us to consider obligation satisfaction as $a_i \in b(\hat{Q}^{(t)}_{X,i}, o_i)$, where

$$b(\hat{Q}^{(t)}_{X,i}, o_i) = \bigcup_{q_{X,i} \in \hat{Q}^{(t)}_{X,i}} b(q_{X,i}, o_i). \tag{5.11}$$

However, the important question now is: how do we learn b such that our behavioral contract is valid?

Learning Contract Components from Data: Since expressions (5.9) define the one-to-one mapping between intentions and options, only the action sets $\mathcal{A}^{exp}(o_i)$, $\mathcal{A}^{reas}(o_i)$ need to be learned. This approach vastly simplifies the learning problem – rather than having to learn a distribution over each agents' actions, it is now only necessary to classify whether an action is "expected," "reasonable," or "feasible." Below, these sets are defined based on intuition and prior knowledge regarding human behavior.

Definition 5. An agent's *reasonable action set* $\mathcal{A}^{reas}(o)$ is the smallest set of actions such that the probability an agent takes an action within this set is greater than $1-\epsilon$:

$$\begin{aligned} \mathcal{A}^{reas}(o) = \ & \arg\min_{C} \quad |\, C \,| \\ & \text{s.t.} \qquad \mathbb{P}(a \in C) > 1-\epsilon. \quad \forall a \in \mathcal{A}. \end{aligned} \tag{5.12}$$

An agent's *expected action set* $\mathcal{A}^{exp}(o)$ is the smallest set of actions such that it is more likely that the agent takes an action within $\mathcal{A}^{exp}(o)$ than outside it:

$$\begin{aligned} \mathcal{A}^{exp}(o) = \ & \arg\min_{C} \quad |\, C \,| \\ & \text{s.t.} \qquad \mathbb{P}(a \in C) > \frac{1}{2} \quad \forall a \in \mathcal{A}. \end{aligned} \tag{5.13}$$

Note that these sets are defined independent of the multiagent context.

Given a discrete action set $\mathcal{A}$, let $a^{(1)}, \ldots, a^{(|\mathcal{A}|)}$ be the possible actions, with cardinality $|\mathcal{A}|$. In order to estimate $\mathcal{A}^{exp}$ and $\mathcal{A}^{reas}$, let us define a probability distribution over the agents' discrete actions as follows:

$$p_i = \mathbb{P}(a_i) \text{ for } i = 1, ..., |\mathcal{A}| \qquad \mathbf{p} = [\ p_1, ..., p_{|\mathcal{A}|}\ ] \qquad \sum_{i=1}^{|\mathcal{A}|} p_i = 1 \tag{5.14}$$

Since $\mathbf{p}$ is uncertain, let us model it as a random variable $\mathbf{P}$ drawn from a generalized Bernoulli distribution. Suppose that associated with a given observation, o, n actions are observed from a dataset $\mathcal{D}$ (i.e. an agent is observed n times at the position/velocity o). Then by considering a uniform Dirichlet distribution as a prior, the posterior distribution is readily computed:

$$\mathbf{P} \sim Dir\Big(\ |\mathcal{A}|\ ,\ \big[\ 1 + \hat{N}_{\mathcal{D}}^{(1)}\ , \ldots,\ 1 + \hat{N}_{\mathcal{D}}^{(|\mathcal{A}|)}\ \big]\ \Big). \tag{5.15}$$

where $\hat{N}_{\mathcal{D}}^{(i)}(o)$ denotes the number of times an agent was observed taking action $a^{(i)}$ given observation o, and Dir represents the Dirichlet distribution. This distribution is also used in the analogous problem of modeling the number of times a dice will come up on different numbers.

Note that $\mathbf{P} \sim \mathbb{P}(\mathbf{p} \mid \mathcal{D})$ fully specifies the action sets $\mathcal{A}^{exp}$, $\mathcal{A}^{reas}$ based on Equations (5.12) and (5.13). Therefore, by drawing several samples of $\mathbf{P}$, one can

compute an under- and over-approximation of the sets $\mathcal{A}^{exp}$, $\mathcal{A}^{reas}$ as follows,

$$\begin{aligned}
\mathcal{A}^{exp}_{lower}(o) &= \{a \in \mathcal{A} \mid a \in \bigcap_{i=1}^{N^{exp}_{\delta}} \mathcal{A}^{exp}(\mathbf{P}_i, o)\} \\
\mathcal{A}^{exp}_{upper}(o) &= \{a \in \mathcal{A} \mid a \in \bigcup_{i=1}^{N^{exp}_{\delta}} \mathcal{A}^{exp}(\mathbf{P}_i, o)\} \\
\mathcal{A}^{reas}_{lower}(o) &= \{a \in \mathcal{A} \mid a \in \bigcap_{i=1}^{N^{reas}_{\delta}} \mathcal{A}^{reas}(\mathbf{P}_i, o)\} \\
\mathcal{A}^{reas}_{upper}(o) &= \{a \in \mathcal{A} \mid a \in \bigcup_{i=1}^{N^{reas}_{\delta}} \mathcal{A}^{reas}(\mathbf{P}_i, o)\}
\end{aligned} \tag{5.16}$$

where $N^{reas}_{\delta} \geq \frac{\log(\delta)}{\log(1-\epsilon)}$ and $N^{exp}_{\delta} \geq \frac{\log(\delta)}{\log(0.5)}$.

Lemma 7. *Given dataset $\mathcal{D}$ of trajectories that are drawn i.i.d. from the underlying distribution of human behavior, the following bounds on $\mathcal{A}^{exp}$ and $\mathcal{A}^{reas}$ hold with probability $1-\delta$:*

$$\begin{aligned}
\mathcal{A}^{exp}_{lower}(\delta) \subseteq \mathcal{A}^{exp} \subseteq \mathcal{A}^{exp}_{upper}(\delta) \\
\mathcal{A}^{reas}_{lower}(\delta) \subseteq \mathcal{A}^{reas} \subseteq \mathcal{A}^{reas}_{upper}(\delta).
\end{aligned} \tag{5.17}$$

From Lemma 7 and the options definitions (5.9), it is possible to obtain analogous probabilistic upper and lower bounds over the obligations b that agents should fulfill, shown below in Equation (5.18). These bound on obligations, placing restrictions over an agent's action set, are depicted in Figure 5.11.

$$b_{lower}\big(q_{i,j}, o_j \mid \delta\big) \;\subseteq\; b\big(q_{i,j}, o_j\big) \;\subseteq\; b_{upper}\big(q_{i,j}, o_j \mid \delta\big). \tag{5.18}$$

Remark 12. Section 5.1 argued the limitations of probabilistic guarantees, so it may seem odd that this section considers probabilistic bounds on the obligations (5.18). However, note that these probabilistic bounds consider randomness in the dataset $\mathcal{D}$, not randomness in human actions (i.e. $\mathcal{D}$ is treated as a random variable rather a_i being treated as a random variable).

Inferring Intention

Once one has computed the (probabilistic) obligations of different agents given their signaled intention, it is necessary to examine how to estimate those intentions.

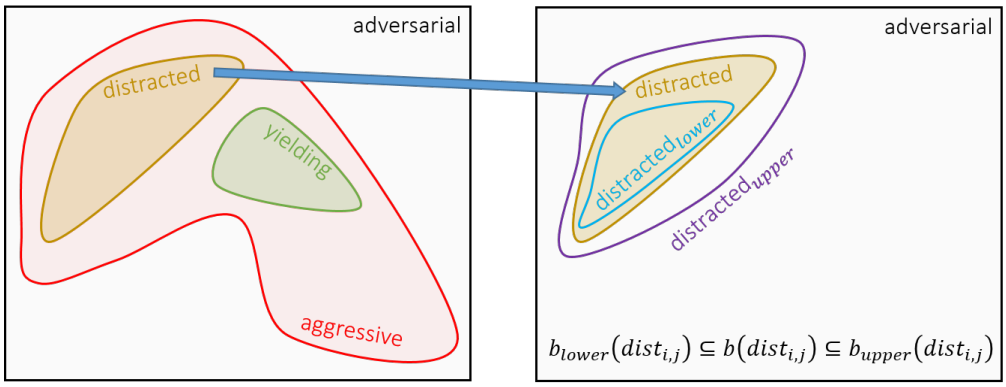


Figure 5.11: *(a)* Diagram showing how each signaled intention is associated with a different part of the action space, which defines the agent's obligation (though agents are not always required to fulfill these obligations). *(b)* Diagram showing probabilistic bounds that lower bound or upper bound each obligation b with a specified probability δ.

Recall that at every time step, agent i has a signaled intention $q_{X,i}^{(t)}$ with respect to Ag_X that it must obey (or *safely* violate). Since $q_{X,i}^{(t)}$ is not explicitly communicated, it must be estimated from agent i's actions.

Let us maintain a belief over $q_{X,i}^{(t)}$, which is denoted by the set $\hat{\mathcal{Q}}_{X,i}^{(t)}$. The goal is to continuously update $\hat{\mathcal{Q}}_{X,i}^{(t)}$ at every time step t, while ensuring that $q_{X,i}^{(t)} \in \hat{\mathcal{Q}}_{X,i}^{(t)}$.

Updating belief over intentions: In the proposed framework, intention is communicated through satisfaction/violation of obligations. For example, an agent signals a change in intention from `distracted` to `aggressive` by taking *unexpected* actions (i.e. violating the obligation associated with a distracted intention) that bring it closer to collision (i.e. satisfying the obligation associated with an aggressive intention). Given this, the following algorithm shows how one can conservatively estimate the belief over intentions $\hat{\mathcal{Q}}_{X,i}$.

- Set $\hat{\mathcal{Q}}_{X,i}^{(t+1)} = \emptyset$
- Observe agent's action a_i
- For $q_{X,i} \in \hat{\mathcal{Q}}_{X,i}^{(t)}$,
  - If $a_i \in b_{upper}(q_{X,i}, o_i)$, then $\hat{\mathcal{Q}}_{X,i}^{(t+1)} = \hat{\mathcal{Q}}_{X,i}^{(t+1)} \cup q_i$
  - If $a_i \notin b^{lower}(o)(q_{X,i}, o_i)$,
    - Compute $\mathcal{Q}_s = \{q \in \mathcal{Q} \mid a_i \in b_{upper}(q_{X,i}, o_i)\ ,\ q \neq q_{X,i}\}$.
    - Obtain $q_+ = \max_{\preceq} \mathcal{Q}_s$ (i.e. the highest-order intention that satisfies the contract).
    - $\hat{\mathcal{Q}}_{X,i}^{(t+1)} = \hat{\mathcal{Q}}_{X,i}^{(t+1)} \cup q_+$

Therefore, after observing a_i, one can update the belief over agent i's signaled intention $\hat{\mathcal{Q}}_{X,i}^{(t+1)}$. In the following section, it is proven that the intention update algorithm outlined above guarantees that $q_{X,i}^{(t+1)} \in \hat{\mathcal{Q}}_{X,i}^{(t+1)}$.

Synthesizing a Controller

The previous sections have defined a mechanism for accurately obtaining signaled intentions $\hat{\mathcal{Q}}_{X,i}^{(t)}$, as well as learning their corresponding obligations $b(\hat{\mathcal{Q}}_{X,i}^{(t)}, o_i)$. Examining the overall planning framework in Figure 5.10, the remaining step is to synthesize a plan for agent X that satisfies its end of the behavioral contract to guarantee system safety.

$$
\begin{aligned}
a_X = \ \arg\min_{a} \quad & J(a, goal) \\
\text{s.t.} \quad & coll(a, b_{upper}(\hat{\mathcal{Q}}_{X,i}^{(t)}, o_i)) = 0 \ \text{ for } \ i = 1, ..., M \\
& coll(a, a_i^{backup}) = 0 \ \text{ for } \ i = 1, ..., M \\
& a \in b(\hat{\mathcal{Q}}_{i,X}^{(t)}, o_X) \quad \text{OR} \quad coll(a, \mathcal{A}_{upper}^{reas}(o_i)) = 0 \\
& a \in \mathcal{A}_{lower}^{reas}(o_X),
\end{aligned}
\tag{5.19}
$$

where a^{backup} is a pre-defined stopping action, and $J(a, goal)$ denotes some cost function encouraging progress to some self-defined *goal*.

All the components described above (learning obligations, inferring intention, synthesizing controller) are put together within our framework as described in Algorithm 3 below and depicted in Figure 5.10. The following section proves the safety of the trajectories generated by this algorithm, and clearly outlines the necessary assumptions that underly the results.

5.2.4 Guaranteeing Safety

The proposed framework centered around the ordered contract automaton, and the derived motion planning algorithm (Algorithm 3), generates safe, efficient motion plans while accounting for other agents' uncertain/changing intentions. The underlying principle is that even though agents may not act cooperatively, their actions are communicative and reflective of intent. For example, agents may take aggressive, unexpected actions, but doing so provides a signal of their intention to others. The main assumptions underlying this framework are summarized as follows:

Assumption 3. *A dataset of agent trajectories, $\mathcal{D}$, is available. The trajectories are drawn i.i.d. from an underlying distribution of human behavior.*

Algorithm 3 Learning Algorithm

Initialize belief over intentions for all agents
(i.e. $\hat{Q}_{X,i}^{(0)} = \{\texttt{distracted}, \texttt{yielding}\}$ for $i = 1, ..., M$).
Measure $o^{(0)} = \left[o_1^{(0)}, ..., o_M^{(0)}\right]$
Compute lower/upper bounds on action sets $\mathcal{A}^{exp}(o_i^{(0)})$, $\mathcal{A}^{reas}(o_i^{(0)})$ from Definition 5 for $i = 1, ..., M$
for t = 1:T **do**
    Observe action that each agent took $a^{(t-1)} = \left[a_1^{(t-1)}, ..., a_M^{(t-1)}\right]$.
    Measure $o^{(t)} = \left[o_1^{(t)}, ..., o_M^{(t)}\right]$
    for i=1:M **do**
        Check obligation satisfaction of $a_i^{(t-1)}$ with respect to $b(q, o^{(t-1)})$ for all $q \in Q$
        Update belief over intentions $\hat{Q}_{X,i}^{(t)}$ from $\hat{Q}_{X,i}^{(t-1)}$ based on intention inference process outlined above
        Re-compute lower/upper bounds on action sets $\mathcal{A}^{exp}(o_i^{(t)})$, $\mathcal{A}^{reas}(o_i^{(t)})$ from Definition 5
        Estimate contractual obligations $b(\hat{Q}_{X,i}^{(t)}, o^{(t)})$ based on options (5.9)
    end for
    Synthesize safe motion plan by solving optimization problem (5.19)
end for

Assumption 4. *Upon initial observation, each agent's signaled intention must be* `distracted` *or* `yielding` *(i.e.* $q_{i,j}^{(0)} \in \{\texttt{distracted}, \texttt{yield}\}$*).*

Assumption 5. *A fixed backup action* a^{backup} *(i.e. braking) is always safe with respect to human agents and other agents' backup action.*

Lemma 8. *If every agent obeys their behavioral contract (see Definition 4), then with probability* $1 - \delta$*,*

$$q_{X,i}^{(t)} \in \hat{Q}_{X,i}^{(t)}, \quad \forall t \in \mathbb{Z}_+ \tag{5.20}$$

In other words, agent i's signaled intention with respect to agent X, $q_{X,i}^{(t)}$*, is a subset of agent X's belief over their signaled intention,* $\hat{Q}_{X,i}^{(t)}$*.*

Proof. This lemma is proved by induction. As a base case, it is know that $q_{X,i}^{(0)} \subseteq \hat{Q}_{X,i}^{(0)}$ by Assumption 4. Assume that $q_{X,i}^{(t)} \in \hat{Q}_{X,i}^{(t)}$. The goal then is to show that $q_{X,i}^{(t+1)} \in \hat{Q}_{X,i}^{(t+1)}$. To do this, let us assume that $q_{X,i}^{(t+1)} \notin \hat{Q}_{X,i}^{(t+1)}$, and show that this results in a contradiction.

Since it is assumed that $q_{X,i}^{(t+1)} \notin \hat{Q}_{X,i}^{(t+1)}$, then let us consider two possibilities: (1) $q_{X,i}^{(t+1)} \in \hat{Q}_{X,i}^{(t)}$, or (2) $q_{X,i}^{(t+1)} \notin \hat{Q}_{X,i}^{(t)}$.

- Suppose $q_{X,i}^{(t+1)} \in \hat{Q}_{X,i}^{(t)}$. Then this means that $q_{X,i}^{(t+1)} \in \hat{Q}_{X,i}^{(t)} \setminus \hat{Q}_{X,i}^{(t+1)}$. By our intention update law, this can occur only if $a^{(t)} \notin b_{upper}(q_{X,i}^{(t+1)}, o_i)$, which means that with probability $1-\delta$, $a^{(t)} \notin b(q_{X,i}^{(t+1)}, o_i)$. However, this action would violate the behavioral contract, leading to a contradiction.

- Suppose $q_{X,i}^{(t+1)} \notin \hat{Q}_{X,i}^{(t)}$. Since $q_{X,i}^{(t+1)} \notin \hat{Q}_{X,i}^{(t)}$ but $q_{X,i}^{(t)} \in \hat{Q}_{X,i}^{(t)}$, then clearly $q_{X,i}^{(t+1)} \neq q_{X,i}^{(t)}$. Based on each agent's behavioral contract, it must hold that $a_i \in b(q_{X,i}^{(t+1)}, o_i) \setminus b(q_{X,i}^{(t)}, o_i)$. Let us consider the following two cases:

  - $q_{X,i}^{(t+1)} \in \hat{Q}_i^{(t)}$: Therefore $a^{(t)} \in b(q_{X,i}^{(t+1)}, o_i)$. By Lemma 7, $a^{(t)} \in b_{upper}(q_{X,i}^{(t+1)}, o_i)$ with probability $1-\delta$. However, this would imply that $q_{X,i}^{(t+1)} \in \hat{Q}_{X,i}^{(t+1)}$, which contradicts our initial assumption.
  - $q_{X,i}^{(t+1)} \notin \hat{Q}_i^{(t)}$: Since $a^{(t)} \in b(q_{X,i}^{(t+1)}, o_i) \setminus a^{(t)} \notin b(q_{X,i}^{(t)}, o_i)$, by Lemma 7, it is known that $a \in b_{upper}(q_{X,i}^{(t+1)}, o_i)$ and $a \notin b_{lower}(q_{X,i}^{(t)}, o_i)$ with probability $1-\delta$. However, by our update law, this only occurs if $q_{X,i}^{(t+1)} \in \hat{Q}_{X,i}^{(t+1)}$, contradicting our initial assumption.

In all cases, a contradiction is reached (that $q_{X,i}^{(t+1)} \in \hat{Q}_{X,i}^{(t+1)}$) with probability at least $1-\delta$, which indicates that our original assumption $q_{X,i}^{(t+1)} \notin \hat{Q}_{X,i}^{(t+1)}$ must be false. Therefore, $q_{X,i}^{(t+1)} \in \hat{Q}_{X,i}^{(t+1)}$. This completes the final step of the induction, and finishes the proof. □

Lemma 9. *The backup action a^{backup} (i.e. stopping) is always safe with respect to other autonomous agents.*

Proof. The controller (5.19) renders every other agent's backup action a^{backup} safe by definition. If (5.19) is infeasible, the agent will take a^{backup}, which is guaranteed safe by Assumption 5. □

Theorem 5. *With probability $1-\delta$, our autonomous agent can have a collision with another agent only if that agent violates its behavioral contract (see Definition 4).*

Proof. Let us denote our agent as agent X and the other agent as agent i. By the definition of the behavioral contracts, at least one of the following two conditions hold under the contract:

- Action $a_i \in b(q_{X,i}^{(t)}, o_i)$,

  - Under this condition, it follows from Lemma 8 that $a_i \in b(q_{X,i}^{(t)}, o_i) \subseteq b(\hat{\mathcal{Q}}_{X,i}^{(t)}, o_i)$ with probability at least $1 - \delta$. Based on Algorithm 3, it is known that either:
    * $coll(a_X, b(\hat{\mathcal{Q}}_{X,i}^{(t)})) = 0$, such that our planned paths do not collide and safety is guaranteed, or
    * the program (5.19) does not have a solution. In this case, agent X can take backup action a_X^{backup}, which by Assumption 5 and Lemma 9 is guaranteed to be safe.

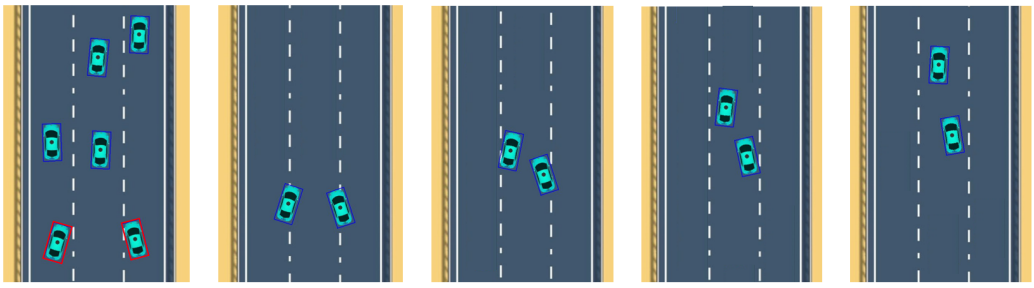
- $coll(a_i, \mathcal{A}^{reas}(o_X)) = 0$ (i.e. no collision with respect to other agents' reasonable actions),
  - Since $\mathcal{A}_{lower}^{reas} \subseteq \mathcal{A}^{reas}$ with probability at least $1 - \delta$, then under this condition, agent X can take any action $a \in \mathcal{A}_{lower}^{reas}$ (or a_X^{backup}) without causing collision. At the next time step, agent i's intention will have been revealed, and it will be subject to a new set of obligations $b(\hat{\mathcal{Q}}_{X,i}^{(t+1)}, o_i)$.

We see that under each of the two conditions (one of which must be satisfied under the behavioral contracts), the controller (5.19) avoids collision with probability $1-\delta$. Therefore, with probability $1 - \delta$, collision can occur only if an agent violates its contract. □

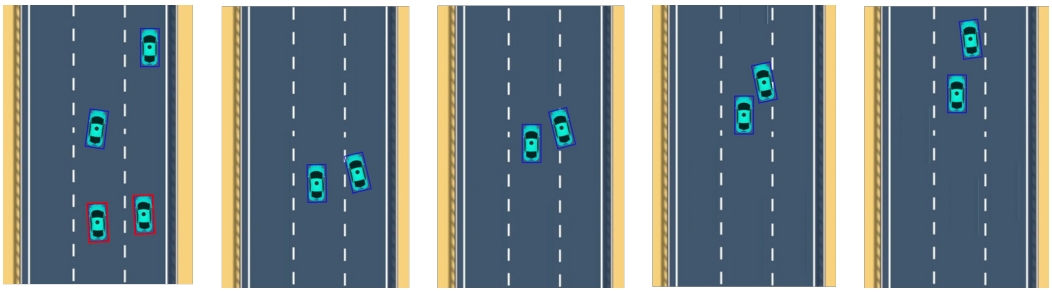
Remark 13. It is important to point out that this framework will not guarantee safety always. Indeed this requirement may be too onerous given the unpredictable nature of human agents. Rather, this framework enables us to say that a collision can occur only if an agent broke its behavioral contract. In most applications, being able to point to such a contract violation as the cause of a collision will be more acceptable than declaring that such a violation occurred due to a low probability event.

5.2.5 Preliminary Results

The proposed safe motion planning algorithm was tested in a simulated highway driving example with multiple agents, as shown in Figure 5.12. Agents were initialized in random lanes with random velocity, and were given a random target lane they would try to safely reach. As agents exited the highway scene (at the top the highway), new agents would be randomly added at the bottom of the highway. In a single randomized trial, the number of agents in the scene was chosen uniformly randomly to be between 2-9, and this number of agents was kept constant throughout



(a) First example instance. Two cars (one in the left lane and one in the right) attempt to merge into the middle lane at a similar time. Because the right agent happens to do its intention inference first, it recognizes the right agent's aggressive intent and slows down, yielding to the left agent. The left agent picks up on this yielding intention during its intention inference update, and merges in front, thus completing this successful negotiation.



(b) Second example instance. The agent on the right begins to merge into the middle lane, and observes a braking action from the agent on its left. Therefore, it infers that the other agent has signaled a yielding intention, and merges in front of it.

Figure 5.12: Top and bottom represent two example instances of simulated cars driving along the highway while running the proposed planner. Each snapshot shows the state of the cars 25 time steps apart, progressing in time from left to right. While the left-most snapshot represents the full multi-agent system, for clarity, only two agents (the ones outlined by the red boxes in the left snapshots) are highlighted in the remaining snapshots. These two agents are used to highlight safe interactive behavior.

the trial. Each trial lasted 1000 time steps (representing approximately 5 highway passes from bottom to top for each agent depending on its velocity).

I first briefly summarize the results, and then provide further details on the different simulation components. To test the algorithm, 100 randomized trials were run, and during those trials, no collisions (i.e. safety violations) occurred. Furthermore, agents were able to reach their target lane over 90% of the time, and were able to safely and successfully negotiate lane changes, even in close proximity to many other agents.

Remark 14. The realized safety of agents in the simulations is encouraging, but should not be surprising given that all agents were designed to satisfy the assump-

tions outlined in Section 5.2.4. Thus, a major test will be whether those assumptions continue to be satisfied if we allow human control of one or more cars.

Action Space: A discrete action space was considered, which was represented as a voxel grid ($a \in \{0,1\}^{21\times21\times6}$), where the first 2 dimensions represent spatial position (x, y) and the 3^{rd} dimension represents time. Therefore, a "1" in a given voxel means that the agent's position was within that grid cell at that moment in the future.

Obligations: Using the trajectory data from dataset $\mathcal{D}$, consisting of observation/action pairs (o, a), the action sets $\mathcal{A}^{exp}$ and $\mathcal{A}^{reas}$ were computed based on Definition 5 with $\epsilon = 0.1$. This yielded several data pairs $(o, \mathcal{A}^{exp})$, $(o, \mathcal{A}^{reas})$, which were used to train separate neural networks using a sigmoidal cross-entropy loss function; these neural networks map observations to these action sets. With $\mathcal{A}^{exp}(o)$ and $\mathcal{A}^{reas}(o)$ learned, these could be used to directly compute agent obligations for the different signaled intentions.

The obligations computed for the different agents provide their potential future actions that must be accounted for. These occupied voxels denote regions that must be avoided in the optimization problem (5.19) (i.e. $b(\hat{\mathcal{Q}}_{X,i}, o_i)$). Note that in our preliminary simulations, probabilistic bounds on the obligations were not computed (i.e. we used $N(\delta) = 1$). Further simulations with probabilistic bounds are left to future work.

Dynamics: The cars in the environment are considered to have control-affine dynamics as described below.

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ v_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + v_t \sin(\theta_t)\Delta t \\ x_t + v_t \cos(\theta_t)\Delta t \\ v_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{5.21}$$

where (x, y) denotes the agent's position, v denotes its velocity, and θ denotes its heading angle. The control inputs are u_1 and u_2. Bounds on velocity and control input are imposed:

$$\begin{aligned} 5 \leq\ & v \leq 16\,, \\ -35 \leq\ & u_1 \leq 15\,, \\ -1 \leq\ & u_2 \leq 1\,. \end{aligned} \tag{5.22}$$

5.3 Conclusion

Section 5.1 in this chapter examined the limitations of currently used uncertainty models in accurately predicting rare events. In the case of driving, it was found that the distribution of human driving trajectories was poorly captured by all these uncertainty models. Even for uncertainty models that do not assume a distribution (e.g. quantile regression), it was shown that the amount of data needed to provide accurate bounds at probability levels required for safety-critical applications is currently infeasible. Therefore, it seems that safety guarantees derived based on these models of uncertainty may only be valid when the probability levels considered are not very low (i.e. $\delta > 0.001$), limiting their application in safety-critical systems.

However, Section 5.2 argues that it is not accurate to model uncertainty in human actions as a random process, and doing so leads to extreme sacrifices in efficiency in order to achieve safety. Instead, certain responsibilities must be imposed on all agents in a multi-agent environment, which bounds the uncertainty in their actions. A framework is introduced that utilizes assume-guarantee contracts to encode agent responsibilities, requiring that agents signal their intention to each other prior to taking dangerous actions and defining certain obligations that they have. Under clearly outlined assumptions, it is proven that the framework guarantees safety provided agents satisfy their contracts, and this is verified through preliminary simulation results. Furthermore, any safety violations can be traced back to specific contract(s) violation.

The preliminary results in Section 5.2 are promising, but they beg the main question: do human agents tend to satisfy the behavioral contracts that have been defined? Therefore, future work will involve computing probabilistic bounds over agent obligations, and testing whether human agents tend to respect the behavioral contracts associated with these obligations.

Appendix D: Fragility of Gaussian Uncertainty Model (Synthetic Data)

The Gaussian uncertainty model was tested on a synthetic 2D data set, using the same process detailed in Section 5.1. Each 2D data point is analogous to a trajectory, $a = \tau_{[2:10]} \sim \mathcal{A}$, in the highD driving dataset. Therefore, the goal is still to learn the model $\hat{F}$ that best matches the data distribution $\mathcal{A}$, minimizing $m(\mathcal{A}||\hat{\mathcal{F}})$. However, using synthetic data allows us to test the accuracy of the uncertainty model with respect to a *known* underlying probability distribution, $\mathcal{A}$.

I randomly generated 10,000 2D points for training data (further increasing the amount of training data did not improve performance) from 3 different distributions: *(a)* perfect Gaussian, *(b)* Gaussian with uniform noise (magnitude of noise was 30% of the data range), and *(c)* Gaussian with symmetric non-uniform noise (also 30% magnitude). For each of these training datasets, the best-fit Gaussian was computed. Then 10,000,000 2D points were generated for the test data *following the exact same distribution as the training data*, and I analyzed how well the computed Gaussian uncertainty model captured the test data.

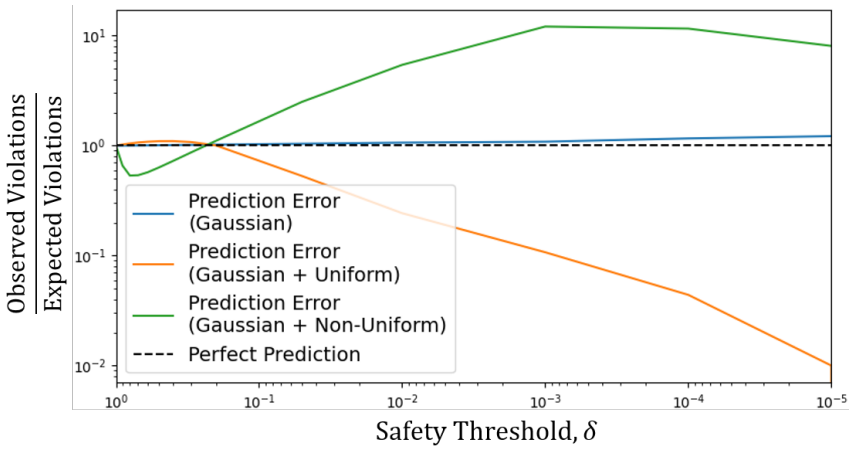


Figure 5.13: Prediction error vs. safety threshold, δ, using a Gaussian uncertainty model on synthetic 2D data generated from 3 different distributions. The dashed black line represents a perfect prediction model. Significant prediction error arises when the underlying data distribution is non-Gaussian.

Figure 5.13 shows that the learned uncertainty model performed very well when the underlying data distribution was Gaussian (blue curve). However, it performed poorly (off by an order-of-magnitude) at low δ when the underlying distribution was non-Gaussian. When the underlying distribution was Gaussian with added uniform noise (orange curve), the observed violations were much lower than the expected violations (i.e. the model was conservative). This is good for safety, but would

clearly lead to overly conservative behavior, especially since the model is off by orders of magnitude.

However, more concerning is the case when the underlying distribution is Gaussian with *non-uniform* noise (green curve). In this case, the observed violations were much higher than the expected violations (greater by an order of magnitude), posing a clear risk for safety-critical applications. This reinforces the results in Section 5.1 by illustrating that significant prediction error inevitably arises, regardless of the amount of training data, when the underlying data distribution is non-Gaussian.

Appendix E: Quantile Regression (Synthetic Data)

The quantile regression experiments from Section 5.1 were repeated using synthetic 2D data rather than real-world driving data. This allowed us to observe how well the uncertainty model performed under ideal conditions when the training/testing data were perfectly i.i.d. I randomly generated 1,000,000 2D training data points (analogous to 1,000,000 trajectories) following a Gaussian distribution, and computed δ-quantile bounds following the same procedure described in Section 5.1 (i.e. computing the smallest convex set containing $1 - \delta$ points). Then 10,000,000 2D test data points were generated *following the exact same distribution as the training data*, and I analyzed how well the computed quantile bounds captured the test data.

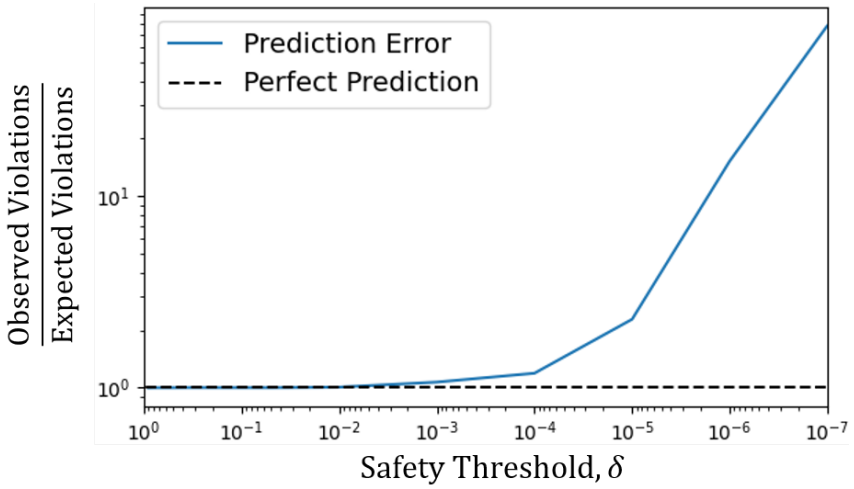
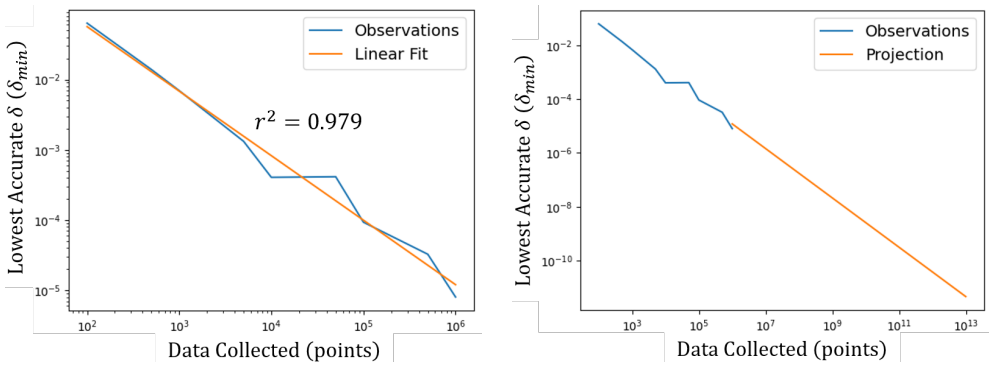


Figure 5.14: Prediction error vs. safety threshold, δ under computed quantile bounds on synthetic 2D data. The dashed black line represents a perfect prediction model.

Figure 5.14 shows the prediction error (i.e. ratio between expected and observed proportion of trajectories outside each quantile) versus the safety threshold δ. The quantile regression model performed very well up to $\delta \geq 0.001$. However, perfor-



(a) (b)

Figure 5.15: *(a)* Smallest accurate δ versus amount of data using synthetic 2D data. The trend is highly linear ($r^2 = 0.979$), *(b)* Projection showing the expected amount of data required to obtain an accurate safety threshold δ_{min}.

mance rapidly deteriorated as δ decreased, meaning the model failed to accurately predict violation probabilities at those safety thresholds.

Using the synthetic data, I computed the smallest accurate safety threshold, δ_{min}, as a function of the amount of training data, N. This threshold δ_{min} was defined as follows,

$$\delta_{min} = \min \delta \quad \text{such that} \quad \left| \log \left(\frac{\text{expected}(\delta)}{\text{observed}(\delta)} \right) \right| \leq \varepsilon \, . \tag{5.23}$$

where $\varepsilon = 0.5$, which represents the vertical distance between the blue curve in Figure 5.14 and the dotted black line. Therefore, δ_{min} represents the smallest δ such that the computed quantile bounds are ε-accurate (as described in Section 3C). Figure 5.15a shows the same inverse linear trend ($\delta_{min} \propto \frac{1}{N}$) on the synthetic data that was seen with the real driving data. Figure 5.15b shows the extrapolation of this trend towards lower δ_{min}. This result reinforces the point made in Section 5.1 that quantile regression can be very accurate for larger δ, but it may not be feasible to collect enough data to reach safety thresholds $\delta_{min} \leq 10^{-8}$.

Chapter 6

CONCLUSION

This thesis has focused on the issue of improving reliability and safety in learning-based controllers, in order to bring them closer to widespread deployment beyond the lab and leverage their impressive performance in solving complex robotics tasks. The central theme across the thesis is that *clearer assumptions* and *limited model information* are crucial to achieving good reliability and assurances of safety for any learning-based controllers.

In Chapter 3, it was shown that integration of model information, in the form of a model-based control prior, into the RL framework effectively constrains the vast policy search space. It is shown, both theoretically and empirically, that constraining this search space greatly reduces variance in learning with minimal bias introduced.

In Chapter 4, it was shown that integration of model information, introduced through discrete-time control barrier functions, into the RL framework effectively constrains the explorable action space to the set of safe agent actions. Uncertainty in the system dynamics, estimated using Gaussian process uncertainty models, can be incorporated into the control barrier functions, providing probabilistic safety guarantees under uncertainty. It is also shown that naive inclusion of a safety filtering mechanism can lead to distortion of the policy gradient in RL, and a new framework is proposed that accounts for this distortion and guides the learning process. In the last section of Chapter 4, the results on discrete-time control barrier functions are extended to multiagent settings with correlated, multivariate uncertainties, thereby showing that the previous safety guarantees can be obtained with a broader class of multiagent settings and a higher-fidelity model of uncertainty.

However, in Section 5.1, widely used uncertainty models were analyzed in depth, and it was found that these uncertainty models can be highly inaccurate when considering probability levels $\delta < 0.001$, casting doubt on their ability to be used in safety-critical applications. Based on these results, I argued that for safety-critical applications dealing with human environments, there is a clear need to shift away from the previous paradigm of learning an uncertain distribution over human actions (treating them as random variables), and designing a controller that is robust to such uncertainty. Treating human actions as random is flawed. Therefore, a

novel framework was proposed in Section 5.2 that imposes loose responsibilities on agents, encoded through assume-guarantee contracts. The components of these assume-guarantee contracts (i.e. obligations) can be learned from data, and can be leveraged to guarantee safety of complex multi-agent systems under contract satisfaction.

6.1 Future Work

Discovering reasonable assume-guarantee contracts: As discussed above, human actions are not random, and in order to balance safety and efficiency in a satisfactory manner, they should not be treated as random. The framework introduced in Section 5.2 attempts to address this issue using behavioral contracts, but there is still significant work to design obligations that human agents respect. In particular, the definitions of "expected" and "reasonable" actions described in Definition 5 are constructed based on prior intuition, and may require modification. Furthermore, tests with human agents must be conducted in order to see if they do indeed tend to satisfy the assume-guarantee contracts that have been constructed.

Leveraging robust control priors: In Chapter 3, a strategy for adapting the regularization term, λ, weighing the control prior vs. the learned controller, was proposed based on the TD-error during learning. However, in many cases, a different adaptive strategy may be appropriate and provide better learning. Furthermore, this work considered a fixed control prior. However, an adaptive control prior may also allow for more sample efficient learning, as data collected online may enable computation of a more optimal/robust control prior. Incorporating an adaptive control prior is not a trivial task though, as updates to the control prior will interfere with learning of the RL controller. Addressing this issue is left for future work.

BIBLIOGRAPHY

[1] URL: https://github.com/rcheng805/CORE-RL/.

[2] URL: https://github.com/rcheng805/RL-CBF/.

[3] URL: https://github.com/rcheng805/robust_cbf/.

[4] Pieter Abbeel, Adam Coates, and Andrew Y. Ng. "Autonomous helicopter aerobatics through apprenticeship learning". In: *International Journal of Robotics Research* (2010). ISSN: 02783649. DOI: 10.1177/0278364910371999. arXiv: 0911.4714 [astro-ph.HE].

[5] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Twenty-first international conference on Machine learning - ICML '04*. 2004. ISBN: 1581138285. DOI: 10.1145/1015330.1015430. arXiv: 1206.5264.

[6] Joshua Achiam et al. "Constrained Policy Optimization". In: *arXiv preprint arXiv:1705:10528* (2017).

[7] Ayush Agrawal and Koushil Sreenath. "Discrete Control Barrier Functions for Safety-Critical Control of Discrete Systems with Application to Bipedal Robot Navigation". In: *Robotics science and systems (RSS)* (2017). ISSN: 2330765X. DOI: 10.15607/RSS.2017.XIII.073.

[8] Mohamadreza Ahmadi et al. "Barrier functions for multiagent-pomdps with dtl specifications". In: *arXiv preprint arXiv:2003.09267* (2020).

[9] Ilge Akkaya et al. "Solving rubik's cube with a robot hand". In: *arXiv preprint arXiv:1910.07113* (2019).

[10] Mohammed Alshiekh et al. "Safe Reinforcement Learning via Shielding". In: *arXiv preprint arXiv:1708.08611* (2017). arXiv: 1708.08611. URL: http://arxiv.org/abs/1708.08611.

[11] Aaron D. Ames, Jessy W. Grizzle, and Paulo Tabuada. "Control barrier function based quadratic programs with application to adaptive cruise control". In: *53rd IEEE Conference on Decision and Control*. 2014. ISBN: 978-1-4673-6090-6. DOI: 10.1109/CDC.2014.7040372.

[12] Aaron D. Ames et al. "Control Barrier Function Based Quadratic Programs for Safety Critical Systems". In: *IEEE Transactions on Automatic Control* (2017). ISSN: 00189286. DOI: 10.1109/TAC.2016.2638961. arXiv: 1609.06408.

[13] Dario Amodei et al. "Concrete problems in AI safety". In: *arXiv preprint arXiv:1606.06565* (2016).

[14] Marcin Andrychowicz et al. "Hindsight experience replay". In: *Advances in neural information processing systems* 30 (2017), pp. 5048–5058.

[15] Georges S. Aoude et al. “Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns”. In: *Autonomous Robots* 35.1 (2013), pp. 51–76.

[16] Sanjeev Arora, Andrej Risteski, and Yi Zhang. “Do GANs learn the distribution? Some Theory and Empirics”. In: *International Conference on Learning Representations*. 2018.

[17] Kai Arulkumaran et al. *Deep reinforcement learning: A brief survey*. 2017. DOI: 10.1109/MSP.2017.2743240. arXiv: 1708.05866.

[18] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. “Action understanding as inverse planning”. In: *Cognition* (2009).

[19] Tirthankar Bandyopadhyay et al. “Intention-aware motion planning”. In: *Springer Tracts in Advanced Robotics*. 2013.

[20] Somil Bansal et al. “Hamilton-jacobi reachability: A brief overview and recent advances”. In: *Conference on Decision and Control, CDC 2017*. ISBN: 9781509028733. DOI: 10.1109/CDC.2017.8263977. arXiv: 1709.07523.

[21] Somil Bansal et al. “MBMF: Model-Based Priors for Model-Free Reinforcement Learning”. In: *arXiv:1709:03153* (2017).

[22] Daman Bareiss and Jur Van Den Berg. “Generalized reciprocal collision avoidance”. In: *International Journal of Robotics Research* (2015). ISSN: 17413176. DOI: 10.1177/0278364915576234.

[23] Federico Bartoli et al. “Context-Aware Trajectory Prediction”. In: *International Conference on Pattern Recognition*. 2018. ISBN: 9781538637883. DOI: 10.1109/ICPR.2018.8545447. arXiv: 1705.02503.

[24] Jonathan Baxter and P.L. Bartlett. “Reinforcement learning in POMDP’s via direct gradient ascent”. In: *International Conference on Machine Learning* (2000).

[25] Felix Berkenkamp et al. “Safe Model-based Reinforcement Learning with Stability Guarantees”. In: *Neural Information Processing Systems (NeurIPS)*. 2017.

[26] Charles Blundell et al. “Weight Uncertainty in Neural Networks”. In: *International Conference on Machine Learning*. Vol. 37. PMLR, July 2015, pp. 1613–1622. arXiv: 1505.05424 [stat.ML].

[27] Urs Borrmann et al. “Control Barrier Certificates for Safe Swarm Behavior”. In: *IFAC Conference on Analysis and Design of Hybrid Systems* (2015). ISSN: 24058963. DOI: 10.1016/j.ifacol.2015.11.154.

[28] Maxime Bouton et al. “Safe reinforcement learning with scene decomposition for navigating complex urban environments”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1469–1476.

[29] Frank Broz, Illah Nourbakhsh, and Reid Simmons. “Planning for Human-Robot Interaction in Socially Situated Tasks: The Impact of Representing Time and Intention”. In: *International Journal of Social Robotics* (2013).

[30] M. C. Campi and S. Garatti. “Wait-and-judge scenario optimization”. In: *Mathematical Programming* (2018).

[31] Ashwin Carvalho et al. “Automated driving: The role of forecasts and uncertainty - A control perspective”. In: *European Journal of Control*. Vol. 24. 2015, pp. 14–32.

[32] G. Cesari et al. “Scenario Model Predictive Control for Lane Change Assistance and Autonomous Driving on Highways”. In: *IEEE Intelligent Transportation Systems Magazine* 9.3 (2017), pp. 23–35.

[33] Changan Chen et al. “Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning”. In: *IEEE ICRA*. 2019.

[34] Mo Chen and Claire J. Tomlin. “Hamilton–Jacobi Reachability: Some Recent Theoretical Advances and Applications in Unmanned Airspace Management”. In: *Annual Review of Control, Robotics, and Autonomous Systems* (2018). ISSN: 2573-5144. DOI: 10.1146/annurev-control-060117-104941.

[35] Yu Fan Chen et al. “Socially aware motion planning with deep reinforcement learning”. In: *IEEE IROS*. 2017.

[36] Yuxiao Chen et al. “Counter-example Guided Learning of Bounds on Environment Behavior”. In: *Proceedings of the Conference on Robot Learning*. Vol. 100. PMLR, Nov. 2020, pp. 898–909.

[37] Richard Cheng et al. “Safe Multi-Agent Interaction through Robust Control Barrier Functions with Learned Uncertainties”. In: *arXiv* (2020).

[38] Yinlam Chow et al. “A Lyapunov-based Approach to Safe Reinforcement Learning”. In: *arXiv preprint arXiv:1805.07708* (2018).

[39] Andrew Clark. “Control barrier functions for complete and incomplete information stochastic systems”. In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 2928–2935.

[40] Stephane Crepey and Matthew F. Dixon. “Gaussian Process Regression for Derivative Portfolio Modeling and Application to CVA Computations”. In: *arXiv* (2019).

[41] Gergely Csibra and György Gergely. “’Obsessed with goals’: Functions and mechanisms of teleological interpretation of actions in humans”. In: *Acta Psychologica* (2007).

[42] Alexander G. Cunningham et al. “MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving”. In: *IEEE ICRA*. 2015.

[43] Richard Dearden, Nir Friedman, and Stuart Russell. “Bayesian Q-learning”. In: *AAAI/IAAI*. 1998, pp. 761–768.

[44] Marc Peter Deisenroth, Carl Edward Rasmussen, and Dieter Fox. “Learning to control a low-cost manipulator using data-efficient reinforcement learning”. In: *Robotics: Science and Systems VII* (2011), pp. 57–64.

[45] Yiannis Demiris. *Prediction of intent in robotics and multi-agent systems*. 2007.

[46] Wenchao Ding, Jing Chen, and Shaojie Shen. “Predicting vehicle behaviors over an extended horizon using behavior interaction network”. In: *International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8634–8640.

[47] Xingping Dong et al. “Hyperparameter optimization for tracking with continuous deep q-learning”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 518–527.

[48] J.C. Doyle et al. “State-space solutions to standard H/sub 2/ and H/sub infinity / control problems”. In: *IEEE Transactions on Automatic ControlTransactions on Automatic Control* (1989). ISSN: 00189286. DOI: 10.1109/9.29425.

[49] John Doyle. “Robust and Optimal Control”. In: *Conference on Decision and Control*. 1996.

[50] Anca D. Dragan. “Robot Planning with Mathematical Models of Human State and Action”. In: *arXiv* (2017).

[51] Yonathan Efroni, Mohammad Ghavamzadeh, and Shie Mannor. “Multi-Step Greedy and Approximate Real Time Dynamic Programming”. In: *arXiv preprint arXiv:1909.04236* (2019).

[52] Andrzej Ehrenfeucht et al. “A general lower bound on the number of examples needed for learning”. In: *Information and Computation* (1989).

[53] D. Ellis, E. Sommerlade, and I. Reid. “Modelling pedestrian trajectory patterns with Gaussian processes”. In: *IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. 2009, pp. 1229–1234.

[54] John J Enright and Peter R Wurman. “Optimization and coordinated autonomy in mobile fulfillment systems”. In: *Workshops at the twenty-fifth AAAI conference on artificial intelligence*. Citeseer. 2011.

[55] Michael Everett, Yu Fan Chen, and Jonathan P. How. “Motion Planning among Dynamic, Decision-Making Agents with Deep Reinforcement Learning”. In: *IEEE IROS*. 2018.

[56] David D Fan et al. “Bayesian Learning-Based Adaptive Control for Safety Critical Systems”. In: *arXiv preprint arXiv:1910.02325* (2019).

[57] David D. Fan, Ali Agha-mohammadi, and Evangelos A. Theodorou. “Deep Learning Tubes for Tube MPC”. In: *arXiv* (2020). arXiv: 2002.01587 [cs.RO].

[58] David D. Fan et al. “Bayesian Learning-Based Adaptive Control for Safety Critical Systems”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 4093–4099.

[59] Farbod Farshidian, Michael Neunert, and Jonas Buchli. “Learning of closed-loop motion control”. In: *IEEE International Conference on Intelligent Robots and Systems*. 2014.

[60] Sarah Ferguson et al. “Real-time predictive modeling and robust avoidance of pedestrians with uncertain, changing intentions”. In: *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*. Springer International Publishing, 2015, pp. 161–177.

[61] Angelos Filos et al. “Can Autonomous Vehicles Identify, Recover From, and Adapt to Distribution Shifts?” In: *International Conference on Machine Learning*. 2020.

[62] Paolo Fiorini and Zvi Shiller. “Motion planning in dynamic environments using velocity obstacles”. In: *International Journal of Robotics Research* (1998). ISSN: 02783649. DOI: 10.1177/027836499801700706.

[63] Jaime Fisac et al. “Probabilistically Safe Robot Planning with Confidence-Based Human Predictions”. In: *Robotics: Science and Systems*. 2018.

[64] Jaime F Fisac et al. “A general safety framework for learning-based control in uncertain robotic systems”. In: *IEEE Transactions on Automatic Control* (2018).

[65] Jaime F. Fisac et al. “A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems”. In: *arXiv preprint arXiv:1705.01292* (2018).

[66] Jaime F. Fisac et al. “Hierarchical game-theoretic planning for autonomous vehicles”. In: *International Conference on Robotics and Automation (ICRA)*. 2019, pp. 9590–9596. DOI: 10.1109/ICRA.2019.8794007.

[67] Jaime F. Fisac et al. “Reach-avoid problems with time-varying dynamics, targets and constraints”. In: *International Conference on Hybrid Systems: Computation and Control, HSCC*. 2015. ISBN: 9781450334334. DOI: 10.1145/2728606.2728612. arXiv: 1410.6445.

[68] Mojtaba Forghani et al. “Design of driver-assist systems under probabilistic safety specifications near stop signs”. In: *IEEE Transactions on Automation Science and Engineering* 13.1 (2016), pp. 43–53.

[69] David Fridovich-Keil et al. “Confidence-aware motion prediction for real-time collision avoidance”. In: *International Journal of Robotics Research* 39.2-3 (Mar. 2020), pp. 250–265. ISSN: 0278-3649. DOI: 10.1177/0278364919859436.

[70] Yarin Gal and Zoubin Ghahramani. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. In: *International Conference on Machine Learning*. 2016.

[71] Javier García and Fernando Fernández. “A Comprehensive Survey on Safe Reinforcement Learning”. In: *Journal of Machine Learning Research* (2015). ISSN: 15337928. DOI: 10.1109/TNNLS.2017.2654539.

[72] Chris Gaskett. “Reinforcement Learning in Circumstances Beyond its Control”. In: *CIMCA*. 2003.

[73] Jin I. Ge et al. “Experimental validation of connected automated vehicle design among human-driven vehicles”. In: *Transportation Research Part C: Emerging Technologies* (2018).

[74] Dibya Ghosh et al. “Divide-and-Conquer Reinforcement Learning”. In: *Neural Information Processing Systems (NeurIPS)*. Vol. abs/1711.09874. 2018.

[75] Tom Gibson. “Recycling Robots”. In: *Mechanical Engineering* 142.01 (Jan. 2020), pp. 32–37. ISSN: 0025-6501. DOI: 10.1115/1.2020-JAN2. URL: https://doi.org/10.1115/1.2020-JAN2.

[76] Jeremy H. Gillula and Claire J. Tomlin. “Guaranteed safe online learning via reachability: Tracking a ground target using a quadrotor”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2012. ISBN: 9781467314039. DOI: 10.1109/ICRA.2012.6225136.

[77] Tobias Gindele, Sebastian Brechtel, and Rüdiger Dillmann. “A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments”. In: *13th International IEEE Conference on Intelligent Transportation Systems*. 2010, pp. 1625–1631.

[78] Andrew Gray et al. “Stochastic predictive control for semi-autonomous vehicles with an uncertain driver model”. In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. 2013, pp. 2329–2334.

[79] Evan Greensmith, Pl Bartlett, and J Baxter. “Variance reduction techniques for gradient estimates in reinforcement learning”. In: *JMLR* (2004).

[80] Shixiang Gu et al. “Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates”. In: *IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 3389–3396.

[81] Agrim Gupta et al. “Social GAN: Socially Acceptable Trajectories With Generative Adversarial Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[82] Arjun K Gupta and Daya K Nagar. *Matrix variate distributions*. Chapman and Hall/CRC, 2018.

[83] Thomas Gurriet et al. "Realizable set invariance conditions for cyber-physical systems". In: *Proceedings of the American Control Conference*. 2019. ISBN: 9781538679265. DOI: 10.23919/acc.2019.8815332.

[84] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *arXiv preprint arXiv:1801.01290* (2018).

[85] Dylan Hadfield-Menell et al. "Cooperative inverse reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2016.

[86] Astghik Hakobyan and Insoon Yang. "Learning-Based Distributionally Robust Motion Control with Gaussian Processes". In: *arXiv* (2020).

[87] Chaozhe R. He, Jin I. Ge, and Gabor Orosz. "Data-based fuel-economy optimization of connected automated trucks in traffic". In: *Annual American Control Conference (ACC)* (2018).

[88] Peter Henderson et al. "Deep Reinforcement Learning that Matters". In: *AAAI Conference on Artificial Intelligence*. 2018.

[89] Yeping Hu, Wei Zhan, and Masayoshi Tomizuka. "Probabilistic Prediction of Vehicle Semantic Intention and Motion". In: *IEEE Intelligent Vehicles Symposium*. Vol. 2018-June. 2018, pp. 307–313.

[90] Riashat Islam et al. "Reproducibility of Benchmarked Deep Reinforcement Learning of Tasks for Continuous Control". In: *Reproducibility in Machine Learning Workshop*. 2017.

[91] Lucas Janson, Tommy Hu, and Marco Pavone. "Safe Motion Planning in Unknown Environments: Optimality Benchmarks and Tractable Policies". In: *Robotics: Science and Systems*. Pittsburgh, USA, June 2018.

[92] Tobias Johannink et al. "Residual Reinforcement Learning for Robot Control". In: *arXiv e-prints*, arXiv:1812.03201 (Dec. 2018), arXiv:1812.03201. arXiv: 1812.03201 [cs.RO].

[93] Gregory Kahn et al. "Uncertainty-Aware Reinforcement Learning for Collision Avoidance". In: *arXiv* (2017).

[94] Dmitry Kalashnikov et al. "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation". In: *arXiv preprint arXiv:1806.10293* (2018).

[95] Richard Kelley et al. "Understanding Human Intentions via Hidden Markov Models in Autonomous Mobile Robots". In: *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction*. Amsterdam, The Netherlands: Association for Computing Machinery, 2008, pp. 367–374. ISBN: 9781605580173. DOI: 10.1145/1349822.1349870.

[96] Hassan K. Khalil. *Nonlinear Systems (Third Edition)*. Prentice Hall, 2000.

[97] Mohammad Javad Khojasteh et al. "Probabilistic safety constraints for learned high relative degree system dynamics". In: *arXiv preprint arXiv:1912.10116* (2019).

[98] Torsten Koller et al. "Learning-based model predictive control for safe exploration". In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE. 2018.

[99] Torsten Koller et al. "Learning-based Model Predictive Control for Safe Exploration and Reinforcement Learning". In: *arXiv preprint arXiv:1803.08287* (2018).

[100] Robert Krajewski et al. "The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems". In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2018.

[101] Matthew Kuipers and Petros Ioannou. "Multiple model adaptive control with mixing". In: *IEEE Transactions on Automatic Control* (2010).

[102] Minae Kwon et al. "When Humans Aren't Optimal: Robots That Collaborate with Risk-Aware Humans". In: *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. Cambridge, United Kingdom: Association for Computing Machinery, 2020, pp. 43–52. ISBN: 9781450367462. DOI: 10.1145/3319502.3374832.

[103] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 6402–6413. arXiv: 1612.01474 [stat.ML].

[104] Chi Pang Lam and S. Shankar Sastry. "A POMDP framework for human-in-the-loop system". In: *IEEE Conference on Decision and Control*. 2014.

[105] Hoang M Le, Cameron Voloshin, and Yisong Yue. "Batch policy learning under constraints". In: *International Conference on Machine Learning*. 2019.

[106] Michelle A Lee et al. "Guided Uncertainty-Aware Policy Optimization: Combining Learning and Model-Based Strategies for Sample-Efficient Policy Learning". In: *arXiv preprint arXiv:2005.10872* (2020).

[107] Sergey Levine and Vladlen Koltun. "Guided Policy Search". In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*. 2013. ISBN: 9781479969227. DOI: 10.1109/ICRA.2015.7138994. arXiv: arXiv:1501.05611v1.

[108] Nan Li et al. "Hierarchical reasoning game theory based approach for evaluation and testing of autonomous vehicle control systems". In: *IEEE Conference on Decision and Control*. 2016.

[109] Timothy P. Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015). ISSN: 1935-8237. DOI: 10.1561/2200000006. arXiv: 1509.02971.

[110] Anqi Liu et al. “Robust Regression for Safe Exploration in Control”. In: vol. 120. PMLR, June 2020, pp. 608–619.

[111] Wei Liu et al. “Situation-aware decision making for autonomous driving on urban road using online POMDP”. In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. 2015, pp. 1126–1133.

[112] Björn Lötjens, Michael Everett, and Jonathan P How. “Safe reinforcement learning with model uncertainty estimates”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8662–8668.

[113] Christos Louizos and Max Welling. “Structured and efficient variational deep learning with matrix gaussian posteriors”. In: *International Conference on Machine Learning*. 2016, pp. 1708–1716.

[114] Wenhao Luo and Ashish Kapoor. “Multi-Robot Collision Avoidance under Uncertainty with Probabilistic Safety Barrier Certificates”. In: *arXiv preprint arXiv:1912.09957* (2019).

[115] Tommaso Mannucci et al. “Safe Exploration Algorithms for Reinforcement Learning Controllers”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2018). ISSN: 21622388. DOI: 10.1109/TNNLS.2017.2654539.

[116] Catharine L.R. McGhan, Ali Nasir, and Ella M. Atkins. “Human intent prediction using Markov decision processes”. In: *Journal of Aerospace Information Systems* (2015).

[117] Rhiannon Michelmore et al. “Uncertainty Quantification with Statistical Guarantees in End-to-End Autonomous Driving Control”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 7344–7350.

[118] Chen Min. “TRUST AND INTENTION IN HUMAN-ROBOT INTERACTION: A POMDP FRAMEWORK”. PhD thesis. National University of Singapore, 2018.

[119] Branka Mirchevska et al. “High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning”. In: *21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 2156–2162.

[120] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[121] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. “A framework for reinforcement learning and planning”. In: *arXiv preprint arXiv:2006.15009* (2020).

[122] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. "Model-based reinforcement learning: A survey". In: *arXiv preprint arXiv:2006.16712* (2020).

[123] Teodor Mihai Moldovan and Pieter Abbeel. "Safe Exploration in Markov Decision Processes". In: *arXiv preprint arXiv:1205.4810* (2012).

[124] Douglas Morrison et al. "Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 7757–7764.

[125] Anusha Nagabandi et al. "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning". In: *arXiv e-prints*, arXiv:1708.02596 (Aug. 2017), arXiv:1708.02596. arXiv: 1708.02596 [cs.LG].

[126] Yashwanth Kumar Nakka et al. "Chance-Constrained Trajectory Optimization for Safe Exploration and Learning of Nonlinear Systems". In: *arXiv* (2020).

[127] *National Highway Traffic Safety Administration. Traffic Safety Facts Annual Report*. 2019.

[128] Quan Nguyen and Koushil Sreenath. "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints". In: *2016 American Control Conference (ACC)*. IEEE. 2016, pp. 322–328.

[129] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. "Local Gaussian Process Regression for Real Time Online Model Learning and Control". In: *Advances in neural information processing systems*. 2009. ISBN: 978-1-4244-2057-5. DOI: 10.1163/016918609X12529286896877.

[130] Haruki Nishimura et al. "Risk-Sensitive Sequential Action Control with Multi-Modal Human Trajectory Forecasting for Safe Crowd-Robot Interaction". In: *arXiv preprint arXiv:2009.05702* (2020).

[131] Motoya Ohnishi et al. "Safety-aware Adaptive Reinforcement Learning with Applications to Brushbot Navigation". In: *arXiv preprint arXiv:1801.09627* (2018).

[132] Kristiaan Pelckmans et al. "Support and Quantile Tubes". In: *arXiv* (2008).

[133] Theodore J Perkins and Andrew G Barto. "Lyapunov design for safe reinforcement learning". In: *Journal of Machine Learning Research* (2003). ISSN: 15324435. DOI: 10.1162/jmlr.2003.3.4-5.803.

[134] Tung Phan-Minh, Karena X Cai, and Richard M Murray. "Towards Assume-Guarantee Profiles for Autonomous Vehicles". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 2788–2795.

[135] Carl Edward Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning*. 2006.

[136] Benjamin Recht. “A Tour of Reinforcement Learning: The View from Continuous Control”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 2.1 (2019), pp. 253–279.

[137] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[138] Amir Sadeghian et al. “SoPhie: An attentive GAN for predicting paths compliant to social and physical constraints”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 1349–1358.

[139] D. Sadigh et al. “Data-driven probabilistic modeling and verification of human driver behavior”. In: *AAAI Spring Symposium*. 2014, pp. 56–61.

[140] Dorsa Sadigh and Ashish Kapoor. “Safe control under uncertainty with Probabilistic Signal Temporal Logic”. In: *Robotics: Science and Systems*. Vol. 12. 2016.

[141] Dorsa Sadigh et al. “Planning for autonomous cars that leverage effects on human actions”. In: *Robotics: Science and Systems*. 2016.

[142] Dorsa Sadigh et al. “Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state”. In: *Autonomous Robots* (2018).

[143] Tim Salzmann et al. “Trajectron++: Multi-Agent Generative Trajectory Forecasting With Heterogeneous Data for Control”. In: *arXiv* (2020). arXiv: 2001.03093.

[144] Hossein Sartipizadeh and Behçet Açıkmeşe. “Approximate convex hull based scenario truncation for chance constrained trajectory optimization”. In: *Automatica* 112 (2020).

[145] Tom Schaul et al. “Prioritized experience replay”. In: *arXiv preprint arXiv:1511.05952* (2015).

[146] John Schulman et al. “High-Dimensional Continuous Control Using Generalized Advantage Estimation”. In: *International Conference on Learning Representations* (2016).

[147] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: *arXiv e-prints*, arXiv:1707.06347 (July 2017), arXiv:1707.06347. arXiv: 1707.06347 [cs.LG].

[148] John Schulman et al. “Trust Region Policy Optimization”. In: *International Conference on Machine Learning (ICML)*. 2015.

[149] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. “On a Formal Model of Safe and Scalable Self-driving Cars”. In: *arXiv* (2017).

[150] David Silver et al. “Deterministic Policy Gradient Algorithms”. In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (2014). ISSN: 1938-7228.

[151] David Silver et al. “Mastering the game of go without human knowledge”. In: *Nature* 550.7676 (2017), pp. 354–359.

[152] Tom Silver et al. “Residual policy learning”. In: *arXiv preprint arXiv:1812.06298* (2018).

[153] Andrew Singletary et al. “Online Active Safety for Robotic Manipulators”. In: 2020. ISBN: 9781728140049. DOI: 10.1109/iros40897.2019.8968231.

[154] Andrew Singletary et al. “Safety-critical rapid aerial exploration of unknown environments”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 10270–10276.

[155] Edward Snelson and Zoubin Ghahramani. “Local and global sparse Gaussian process approximations”. In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)* (2007). ISSN: 15324435.

[156] Mohit Srinivasan, Nak-seung P Hyun, and Samuel Coogan. “Weighted Polar Finite Time Control Barrier Functions With Applications To Multi-Robot Systems”. In: *IEEE Conference on Decision and Control*. 2019.

[157] Mohit Srinivasan et al. “Synthesis of Control Barrier Functions using a Supervised Machine Learning Approach”. In: *arXiv* (2020).

[158] Shengyang Sun, Changyou Chen, and Lawrence Carin. “Learning Structured Weight Uncertainty in Bayesian Neural Networks”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2017, pp. 1283–1292.

[159] Richard Sutton et al. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: *Advances in Neural Information Processing Systems* (1999). arXiv: 1609.06838.

[160] Richard S Sutton. “First results with Dyna, an integrated architecture for learning, planning and reacting”. In: *Neural networks for control* 179 (1990).

[161] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

[162] Natasa Tagasovska and David Lopez-Paz. “Single-Model Uncertainties for Deep Learning”. In: *Neural Information Processing Systems*. 2018.

[163] Jie Tang et al. “Parameterized maneuver learning for autonomous helicopter flight”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2010. ISBN: 9781424450381. DOI: 10.1109/ROBOT.2010.5509832.

[164] Duy Tran et al. “A Hidden Markov Model based driver intention prediction system”. In: *IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. 2015, pp. 115–120.

[165] Pete Trautman et al. “Robot navigation in dense human crowds: Statistical models and experimental studies of human-robot cooperation”. In: *The International Journal of Robotics Research* 34.3 (2015), pp. 335–356. DOI: 10.1177/0278364914557874.

[166] Peter Trautman and Andreas Krause. “Unfreezing the robot: Navigation in dense, interacting crowds”. In: *IEEE IROS*. 2010.

[167] Jur Den Van Berg, Ming Lin, and Dinesh Manocha. “Reciprocal velocity obstacles for real-time multi-agent navigation”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2008. ISBN: 9781424416479. DOI: 10.1109/ROBOT.2008.4543489.

[168] Abhinav Verma et al. “Programmatically Interpretable Reinforcement Learning”. In: *International Conference on Machine Learning (ICML)*. 2018.

[169] Kim P Wabersich and Melanie N Zeilinger. “Safe exploration of nonlinear dynamical systems: A predictive safety filter for reinforcement learning”. In: *arXiv preprint arXiv:1812.05506* (2018).

[170] Akifumi Wachi et al. “Safe Exploration and Optimization of Constrained MDPs using Gaussian Processes”. In: *32nd AAAI conference on Artificial Intelligence (AAAI)* (2018).

[171] Niklas Wahlström, Thomas B Schön, and Marc Peter Deisenroth. “From pixels to torques: Policy learning with deep dynamical models”. In: *arXiv preprint arXiv:1502.02251* (2015).

[172] Li Wang, Aaron Ames, and Magnus Egerstedt. “Safety barrier certificates for heterogeneous multi-robot systems”. In: *Proceedings of the American Control Conference*. 2016. ISBN: 9781467386821. DOI: 10.1109/ACC.2016.7526486. arXiv: 1609.00651.

[173] Li Wang, Evangelos A Theodorou, and Magnus Egerstedt. “Safe learning of quadrotor dynamics using barrier certificates”. In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2018.

[174] Mingyu Wang et al. “Safe Distributed Lane Change Maneuvers for Multiple Autonomous Vehicles Using Buffered Input Cells”. In: *International Conference on Robotics and Automation*. 2018. ISBN: 9781538630815. DOI: 10.1109/ICRA.2018.8460898.

[175] *Waymo Safety Report*. Waymo. 2020.

[176] Lex Weaver and Nigel Tao. “The Optimal Reward Baseline for Gradient-Based Reinforcement Learning”. In: *Uncertainty in Artificial Intelligence (UAI)*. 2001.

[177] David Wilkie, Jur Van Den Berg, and Dinesh Manocha. "Generalized velocity obstacles". In: *International Conference on Intelligent Robots and Systems*. 2009. ISBN: 9781424438044. DOI: 10.1109/IROS.2009.5354175.

[178] Ronald J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning* (1992).

[179] Cathy Wu et al. "Variance Reduction for Policy Gradient with Action-Dependent Factorized Baselines". In: *International Conference on Learning Representations*. 2018.

[180] Bernhard Wymann et al. *TORCS, The Open Racing Car Simulator*. 2014.

[181] Wenda Xu et al. "Motion planning under uncertainty for on-road autonomous driving". In: *IEEE ICRA*. 2014.

[182] Guang Yang, Calin Belta, and Roberto Tron. "Self-triggered control for safety critical systems using control barrier functions". In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 4454–4459.

[183] Je Hong Yoo and Reza Langari. "A stackelberg game theoretic driver model for merging". In: *ASME Dynamic Systems and Control Conference (DSCC)*. 2013.

[184] Marvin Zhang et al. "Solar: Deep structured representations for model-based reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7444–7453.

[185] Wenbo Zhang, Osbert Bastani, and Vijay Kumar. "MAMPS: Safe Multi-Agent Reinforcement Learning via Model Predictive Shielding". In: *arXiv e-prints*, arXiv:1910.12639 (Oct. 2019), arXiv:1910.12639. arXiv: 1910.12639 [eess.SY].

[186] Tingting Zhao et al. "Analysis and improvement of policy gradient estimation". In: *Neural Networks* (2012).

[187] Dingjiang Zhou et al. "Fast, On-line Collision Avoidance for Dynamic Vehicles Using Buffered Voronoi Cells". In: *IEEE Robotics and Automation Letters* (2017). ISSN: 23773766. DOI: 10.1109/LRA.2017.2656241.

[188] Brian D. Ziebart et al. "Maximum entropy inverse reinforcement learning". In: *Proceedings of the National Conference on Artificial Intelligence*. Vol. 3. 2008, pp. 1433–1438.